A.1 INPUT DATA FILE FORMAT

To expedite input process, SIMCOL uses data files to input structural design data and collision scenarios. The format of input data files of SIMCOL Version 2.1 is given in the following sections.

A.1.1 Structural Data (structure.in)

A.1.1.1 Single Hull Tanker

(principle dimensions)

SHIP_TYPE (1 for single hull)

LBP1 B1 D1 T1 DISP1 WEB_SPC (struck ship)

LBP2 B2 T2 DISP2 BOWHT HEA (striking ship)

(transverse bulkheads)

TBHD_NUM(<12)
TBHD_LOC(1) (2) ... (TBHD_NUM)

(longitudinal bulkheads)

LBHD_NUM(<7)
LBHD_LOC(1) LBHD_THK(1) MAT_GR(1)
...
LBHD_LOC(N/2) LBHD_THK(N/2) LBHD_MAT(N/2)

(decks, bottoms and stringers)

DECK_THK BTM_THK
STRG_NUM(<5) STRG_WID
STRG_LOC(1) (2) ... (STRG_NUM)
STRG_THK(1) (2) ... (STRG_NUM)

(web frames)

WEB_SUP(1)(<4) WEB_DEP(1) WEB_STF(1)
WEB_MAT(1,1) WEB_THK(1,1) WEB_SMZ(1,1)
...
WEB_MAT(1,S) WEB_THK(1,S) WEB_SMZ(1,S)
WEB_SPN(1,1) ... WEB_SPN(1,S-1)
WEB_GAP(2,1) ... WEB_SPN(1,S-1)
SUP_MAT(1,2) SUP_ARA(1,2) SUP_GRA(1,2) SUP_LEN(1,2)
...
SUP_MAT(1,S-1) SUP_ARA(1,S-1) SUP_GRA(1,S-1) SUP_LEN(1,S-1)
STF_THK(1) STF_GRA(1)
******************************************************************************
...(repeating web supports)
******************************************************************************
### A.1.1.2 Double Hull Tanker

---

**Principle dimensions**

- **SHIP_TYPE**: 2 for double hull

---

**Transverse bulkheads**

- **TBHD_NUM**: <=12
- **TBHD_LOC**: (1) (2) ... (TBHD_NUM)

---

**Longitudinal bulkheads**

- **LBHD_NUM**: <=7
- **LBHD_LOC**: (1) **LBHD_THK**: (1) **MAT_GR**: (1)
- **LBHD_LOC**: (N/2) **LBHD_THK**: (N/2) **LBHD_MAT**: (N/2)

---

**Decks, bottoms and stringers**

- **DECK_THK**: IBTM_THK **BTM_THK**: DBL_HT
- **STRG_NUM**: <=5
- **STRG_LOC**: (1) (2) ... (STRG_NUM)
- **STRG_THK**: (1) (2) ... (STRG_NUM)

---

**Web frames**

- **WEB_SUP**: (2) (<4) **WEB_DEP**: (N/2) **WEB_STF**: (N/2)
- **WEB_MAT**: (2,1) **WEB_THK**: (2,1) **WEB_SMZ**: (2,1)
- **WEB_MAT**: (2,S) **WEB_THK**: (2,S) **WEB_SMZ**: (2,S)
- **WEB_SPN**: (2,1) ... **WEB_SPN**: (1,S-1)
- **WEB_GAP**: (2,1) ... **WEB_SPN**: (1,S-1)
- **SUP_MAT**: (2,2) **SUP_ARA**: (2,2) **SUP_GRA**: (2,2) **SUP_LEN**: (2,2)
- **SUP_MAT**: (2,S-1) **SUP_ARA**: (2,S-1) **SUP_GRA**: (2,S-1) **SUP_LEN**: (2,S-1)
- **STF_THK**: (2) **STF_GRA**: (2)

---

*Note: Properties of side shell webs [(1) or (1,?)] are identical to those of inner skin webs [(2) or (2,?)].*
### A.1.2 Collision Scenarios (collision.in)

<table>
<thead>
<tr>
<th>Number of collision scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
</tr>
</tbody>
</table>
| ...

... (repeating scenarios)
A.2 FLOWCHARTS OF MAJOR ROUTINES

The following figures (Figures A.1 through A.6) contain flowcharts of major routines in SIMCOL Version 2.1.

![Flowchart of Major Routines](image)

Figure A.1 MAIN
Initiate data of ship structure & collision scenario, INI_COLLIS

Initiate variables; Calculate virtual mass, time step, orientation of ships and relationship to \( \xi-\eta \) system

Is the relative movement marginal?

Start a new time step:
Calculate relative transition, initial kinetic energy, movement of impact point in the struck ship

Have stroke through the cargo block?

Calculate orientation of ships, shape of penetrated bow, damaged extents and virtual mass

Calculate Minorsky forces, MINO_FORCE, and accelerations caused by Minorsky forces

Determine number of reached bulkheads. Calculate membrane forces, MEMB_FORCE, and accelerations caused by membrane forces

Determine accelerations, velocities, locations and orientations of ships, and kinetic energy lost in the time step; Output detail parameters

Is force or lost kinetic energy negligible?

Output summary and return to MAIN

Figure A.2 COLLISION
COLLISION

Calculate Minosky forces from Decks, MINO_DECK

Are there any stringers?

Yes

Calculate Minosky forces from stringers, MINO_DECK

No

Oblique collision?

Yes

Calculate Minosky forces from side and longitudinal bulkheads, MINO_SIDE

No

COLLISION

Calculate total forces

Figure A.3 MINO_FORCE

COLLISION

Initiate variables, set DH flags

MEMB_FORCE (b)

For every penetrated longitudinal bulkheads

Yes

Calculate load capacity of webs, LD_FCT, determine impact location relative to webs & transverse bulkheads

No

For every web been directly pushed by bow

Yes

Calculate web resistant forces and absorbed energy

Figure A.4(a) MEMB_FORCE (a)
For every reached longitudinal bulkheads

Determine effective breadth, bending angle at top and bottom, membrane stress, impact location relative to webs and transverse bulkheads

Has bending angle exceeded the limit?

Yes

Ruptured

No

Has the striking bow swept across web or bulkhead?

Yes

Oblique collision?

Yes

Calculate propagating force

No

No

Calculate load on webs and web supporting capacity, LD_FCT

Strike on web directly?

Yes

Calculate web resistant force and absorbed energy

No

Estimate and verify crushed web number, determine the allowable deflection, DEFL_LIM

Has the deflection limit been exceeded?

Yes

Ruptured

No

MEMB_FORCE (c)

MEMB_FORCE (a)

Determine the absorbed energy during the time step, calculate total forces, reset variables

COLLISION

MEMB_FORCE (c)
For every web spaces within the damage length

Calculate deflection at web

Oblique collision?

Yes

Calculate web resisting force

No

Calculate web absorbed energy and elongation in this web space

Double Hull?

Yes

Calculate elongation of inner skin, bending angle of inner skin

Has the bending angle exceeded the limit?

Yes

Ruptured

No

Calculation the total energy absorbed till this time step

Figure A.4(c) MEMB_FORCE (c)
MINO_FORCE

Initiate variables

For every bow segment

Yes

Determine segment vector and their movement vectors

Based on the movement vectors determine nodes of swept area, calculate the area, the area center, AR_CTR, and the average movement of nodes

Calculate Minosky forces from side and longitudinal bulkheads, MINO_SIDE

No

MINO_FORCE

Calculate total forces and absorbed energy in this time step

Figure A.5 MINO_DECK

MINO_FORCE

Initiate variables, calculate longitudinal bow movement at tip and at side shell

For every ruptured longitudinal bulkhead

No

Yes

Determine bow movement at this longitudinal bulkhead

Calculate the resisting force and absorbed energy

Figure A.6 MINO_SIDE
A.3 FORTRAN SOURCE CODE

The following sections are the FORTRAN source code of SIMCOL Version 2.1, including main.for, module.for, collision_mod.for, initiate.for and collision.for.

A.3.1 main.for

```fortran
!*** MAIN.FOR   by Donghui Chen
!
!** VERSION: 2.1  DATE: December 10, 1999
!
!** REVISION NOTES:
! v2.1
! - Incorporate definite bow depth
! v2.0
! - Incorporate Double Hull Tankers
! v1.2
! - INPUT & OUTPUT
!   Rearrange file open procedure.
! v1.1
! - INPUT
!   STRUCTURE.IN - data of structure and arrangement
!   COLLISION.IN - data of collision scenario
! v1.0
! - INPUT
!   - Allow multiple runs.
!   - All input data is given by input.dat.
! - OUTPUT
!   - Set damage.dat as a output of summery.
!
!** CONTENTS:
! - PROGRAM
!  MAIN - the main controlling procedure for the collision analysis
!
PROGRAM MAIN
!
IMPLICIT NONE
!
INTEGER :: STAT, I, N
!
! Open input file
OPEN (UNIT=02, FILE="collision.in")
!
! Open output files
OPEN (UNIT=21, FILE="damage.out")
OPEN (UNIT=22, FILE="simulation.out")
OPEN (UNIT=23, FILE="energy.out")
OPEN (UNIT=24, FILE="rupture.out")
!
WRITE (21,'(A,/)') "DAMAGE.OUT v2.1"
WRITE (21,*) "     INI LOC     INI PHI     MAX PEN     DAM LEN"
WRITE (22,'(A,/)') "SIMULATION.OUT v2.1"
```

WRITE (22,'(A88)') "No PEN D_PEN LOC"
& /* D_LOC REL_TRANS ZETA"
WRITE (22,'(A88)') "VX1 VY1 W1 VX2"
& /* VY2 W2"
WRITE (22,'(A88)') "X1 Y1 OMEGA1 X2"
& /* Y2 OMEGA2"
WRITE (23,'(A,/)') "ENERGY.OUT v2.1"
WRITE (24,'(A,/)') "RUPTURE.OUT v2.1"
WRITE (24,*') " INI LOC INI PHI OUTER INNER"
!
READ (02,*) N
!
CALL INI_SHIP
!
Start collision simulation
DO I=1, N
 CALL COLLISION(1.E-5, STAT, N)
 IF (N<=2) THEN
 SELECT CASE (STAT)
 CASE (0)
 PRINT *, "Get together."
 CASE (1)
 PRINT *, "Get out of cargo block."
 CASE (2)
 PRINT *, "Cut the struck ship in half."
 CASE (3)
 PRINT *, "Seperated."
 CASE DEFAULT
 PRINT *, "Something wrong."
 END SELECT
 END IF
 END DO
!
CLOSE (02)
CLOSE (21)
CLOSE (22)
CLOSE (23)
CLOSE (24)
!
END PROGRAM MAIN

A.3.2 module.for

!!! MODULE.FOR by Donghui Chen
!!
!!! VERSION: 2.1 DATE: December 10, 1999
!!
!!! REVISION NOTES:
!!! v2.1
!!! - Incorporate definite bow depth
!!! v2.0
!!! - Incorporate Double Hull Tankers
!!! - Introduce SHIP_TYPE into SHIP_STR
!!! v1.2
!!! - Introduce properties of web frames into module SHIP_STR
!!! v1.1
!!! - Separate out modules SHIP_COL and COL_FORCE to COLLISION_MOD.FOR
!!! - Introduce material grade, 1 - MS, 2 - HT32, 3 - HT36 in SHIP_STR
**v1.0**

- **MEMBRANE EFFECTS**
  - Let upper level procedure determine the effective breadth, for which the depth of struck ship is suggested.
  - It is suggested to use smeared plating thickness.
  - Revised the formula used to calculate the membrane forces.

- **MINORSKY MECHANISM**
  - Revised method to calculate damaged area.

- **VIRTUAL MASS**
  - Correct the formula (Rev 02/17)
  - Add one variable for determinant (Rev 03/04)

** CONTENTS:**

- **MODULES**
  - GENERAL - defines some general parameters and functions
  - SHIP_DIM - defines variables of ship dimensions
  - SHIP_STR - defines variables of ship structural details

** Module GENERAL is to define some general parameters and functions used in calculations **

** CONTENTS:**

- **PARAMETERS**
- **FUNCTIONS**
  - V_ABS - calculates the absolute value of vector
  - INTCPT - calculates the interception point of two lines
- **SUBROUTINES**
  - AR_CTR - calculates area and shape center of a polygon

** MODULE GENERAL **

* Units:  dimensions - (meters)
  - mass/displacement - (kilograms)
  - angles - (degrees)
  - velocity - (meters/second) [x KTM to transform from knots]
  - (degrees/second)
  - time - (seconds)
  - stress - (Pascals)
  - force - (Newtons)

REAL, PARAMETER :: RHO=1025., PI=3.1415926, KTM=0.5143,
& EM=2.08E11, PR=.3, RTD=180./PI

The following part contains the relevant functions CONTAINS

* Function V_ABS is to calculate the absolute value of a vector
Input : A - a vector of any dimension
D - dimension of A if A is not a 2-dimensional vector, OPTIONAL
Output: V_ABS - the absolute value

REAL FUNCTION V_ABS(A, D)
IMPLICIT NONE
REAL, DIMENSION(2), INTENT(IN) :: A
INTEGER, OPTIONAL :: D
INTEGER :: I
IF (PRESENT(D)) THEN
  V_ABS=0
  DO I=1, D
    V_ABS=V_ABS+A(I)**A(I)
  END DO
ENDIF
V_ABS = SQRT(V_ABS)
ELSE
   V_ABS = SQRT(A(1)*A(1) + A(2)*A(2))
END IF
END FUNCTION V_ABS

!* Function INTCPT is to calculate the interception point of two lines
!* Input : X11 - the first point of the first line
!*   X12 - the second point of the first line
!*   X21 - the first point of the second line
!*   X22 - the second point of the second line
!* Output: INTPT - the interception point
!* 
FUNCTION INTCPT(X11, X12, X21, X22) RESULT(INTPT)
IMPLICIT NONE
REAL, DIMENSION (2), INTENT (IN) :: X11, X12, X21, X22
REAL, DIMENSION (2) :: INTPT
REAL :: D
D = (X11(1) - X12(1))*(X21(2) - X22(2)) - (X21(1) - X22(1))*(X11(2) - X12(2))
INTPT(1) = -(X11(1) - X12(1))*(X21(1)*X22(2) - X22(1)*X21(2))
&   + (X21(1) - X22(1))*(X11(1)*X12(2) - X12(1)*X11(2))
INTPT(2) = -(X11(2) - X12(2))*(X21(1)*X22(2) - X22(1)*X21(2))
&   + (X21(2) - X22(2))*(X11(1)*X12(2) - X12(1)*X11(2))
IF (D == 0.) THEN
   INTPT = 0.
ELSE
   INTPT = INTPT / D
END IF
END FUNCTION INTCPT

!* Function INTCPTP is to calculate the interception point of a line with y
!* Input : X11 - the first point of the first line
!*   X12 - the second point of the first line
!*   Y - y value of the second line
!* Output: INTPT - the interception point
!* 
FUNCTION INTCPTP(X11, X12, Y) RESULT(INTPT)
IMPLICIT NONE
REAL, DIMENSION (2), INTENT (IN) :: X11, X12
REAL, INTENT (IN) :: Y
REAL, DIMENSION (2) :: INTPT
REAL :: D
INTPT(1) = X11(1) + (X12(1) - X11(1))*Y/(X12(2) - X11(2))
INTPT(2) = Y
END FUNCTION INTCPTP

!* Subroutine AR_CTR is to calculate area and shape center of a polygon
!* Input : NM - number of nodes, maximum 4
!*   PT - node coordinates, in clock-wise direction
!*   WD - y of excluded strip
!* Output: AREA - area of the polygon
!*   CTR - shape center of the polygon
!* 
SUBROUTINE AR_CTR(NM, PT, AREA, CTR, WD)
IMPLICIT NONE
INTEGER, INTENT (IN) :: NM
REAL, DIMENSION (4,2), INTENT (IN) :: PT
REAL, DIMENSION (2), OPTIONAL, INTENT (IN) :: WD
REAL, INTENT (OUT) :: AREA
REAL, DIMENSION (2), INTENT (OUT) :: CTR
!
PTX - peripheral points of excluded portion
AREAX - area of excluded portion
** Module SHIP_DIM defines variables of ship dimensions and functions used to calculate the virtual mass and mass moment of inertia.

** CONTENTS:
- COMMON BLOCKS
  STRUCK_DIM
  STRIKE_DIM
- FUNCTIONS
  ANN - calculates the added mass tensor on ship's principal axis
  MASS-VIRT - calculates the virtual mass
  J-VIRT - calculates the virtual mass moment of inertia

MODULE SHIP_DIM

IMPLICIT NONE

!* Variables of struck ship's dimensions
!* REAL :: LBP1, BEAM1, DEPTH1, DRAFT1, DISP1
!* A1 - added mass on the principal axis, i.e. all and a22
!* JV1 - virtual mass moment of inertia
!* REAL, DIMENSION (2) :: A1
!* REAL :: JV1
!* COMMON /STRUCK_DIM/ LBP1, BEAM1, DEPTH1, DRAFT1, DISP1, A1, JV1
!* Variables of striking ship's dimensions
!* HEA - half-entrance angle of bow
!* REAL :: LBP2, BEAM2, DRAFT2, DISP2, BOW_HT, HEA
!* A2 - added mass on the principal axis, i.e. all and a22
!* JV2 - virtual mass moment of inertia
!* REAL, DIMENSION (2) :: A2
!* REAL :: JV2
!* COMMON /STRIKE_DIM/ LBP2, BEAM2, DRAFT2, DISP2, BOW_HT, HEA,
* & A2, JV2
!* MV - virtual mass of the ship (Rev 03/04)
*! where MV(1) - m11
*! MV(2) - m22
*! MV(3) - m12 and m21
*! MV(4) - determinant of mass matrix
!* REAL, DIMENSION (4) :: MV1, MV2  ! 1 - struck ship; 2 - striking ship
*! The following part contains the relevant functions CONTAINS
!* Function ANN is to calculate the added-mass tensor on the principal axis,
!* i.e. all and a22.
!* Input : BEAM, DRAFT, DISP, LBP
!* Output: BNN - all and a22
!* FUNCTION ANN(BEAM, DRAFT, DISP, LBP) RESULT(BNN)
USE GENERAL, ONLY : RHO, PI
REAL, INTENT (IN) :: BEAM, DRAFT, DISP, LBP
REAL, DIMENSION (2) :: BNN
IF(DISP<=0) THEN
    BNN(1)=0
    BNN(2)=0
ELSE
    BNN(1)=0.75225*RHO*(BEAM*DRAFT)**1.5/DISP
    BNN(2)=1.189*RHO*DRAFT*DRAFT*LBP/DISP
END IF
END FUNCTION ANN

!* Function MASS_VIRT is to calculate the virtual mass of ships (Rev 02/17)
! Input : AN - a11 and a22, THETA, DISP
! Output: MV - virtual mass m11/m22 and m12/m21
!
FUNCTION MASS_VIRT(AN, THETA, DISP) RESULT(MV)
REAL, INTENT (IN) :: AN(2), THETA, DISP
REAL, DIMENSION (4) :: MV
MV(1)=(AN(1)*(COSD(THETA))**2+AN(2)*(SIND(THETA))**2+1.)*DISP
MV(2)=(AN(1)*(SIND(THETA))**2+AN(2)*(COSD(THETA))**2+1.)*DISP
MV(3)=(AN(1)-AN(2))*SIND(THETA)*COSD(THETA)*DISP
MV(4)=MV(1)*MV(2)-MV(3)*MV(3)
END FUNCTION MASS_VIRT

!* Function J_VIRT is to calculate the virtual mass moment of inertia
! Input : LBP, DRAFT, DISP
! Output: J_VIRT - virtual mass moment of inertia
!
REAL FUNCTION J_VIRT(LBP, DRAFT, DISP)
USE GENERAL, ONLY : RHO
REAL, INTENT (IN) :: LBP, DRAFT, DISP
J_VIRT=DISP*LBP*LBP/12.*1.21
END FUNCTION J_VIRT

END MODULE SHIP_DIM

!*** Module SHIP_STR defines variables of ship structural details
!
CONTENTS:
- COMMON BLOCK
  CONSTRUCTION
  WEBS
- FUNCTIONS
  LOAD - calculate load pattern on each span
  A_BND - calculates allowable bending load at each support
  A_SHR - calculates allowable shear load at each support
  A_CMP - calculates allowable compression load at each support
  A_CRS - calculates allowable cruching load
  LD_FCT - calculates actual loads vs allowable loads of webs

MODULE SHIP_STR
IMPLICIT NONE
!* Variables of ship construction
SHIP_TYPE - type of tanker constructions
  1: single hull; 2: double hull; and 3: IOTD
TBHD_NUM - number of transverse bulkheads in the cargo block
LBHD_NUM - number of longitudinal bulkheads plus side shell
! TBHD_LOC - the location of transverse bulkheads (from midship)
! LBHD_LOC - the location of side shell and longitudinal bulkheads
! LBHD_THK - plate thickness of side shell and each longitudinal bulkhead
! LBHD_MAT - material grade of bulkheads and shell plating
! LBHD_RUP - flag for rupture, 1 - ruptured (top/btm); 2 - ruptured totally
!
! WEB_SPC - side web spacing in cargo block
!
! DECK_THK - plating thickness of decks
! IBTM_THK - plating thickness of inner bottom
! BTM_THK - plating thickness of bottom
! DBL_HT - double bottom height
! TDAB_THK - total plating thickness of decks and bottoms
!
! STRG_NUM - number of stringers
! STRG_WID - stringer width
! STRG_LOC - stringer location measured from baseline
! STRG_THK - stringer thickness
! TSTR_THK - total thickness of affected stringers
!
 INTEGER :: SHIP_TYPE, TBHD_NUM, LBHD_NUM, STRG_NUM
!
 REAL, DIMENSION (12) :: TBHD_LOC
 REAL, DIMENSION (7) :: LBHD_LOC
 REAL, DIMENSION (4) :: LBHD_THK
 INTEGER, DIMENSION (4) :: LBHD_MAT
 INTEGER, DIMENSION (7) :: LBHD_RUP
!
 REAL :: WEB_SPC, DECK_THK, IBTM_THK, BTM_THK, DBL_HT, TDAB_THK
 REAL :: STRG_WID, TSTR_THK
 REAL, DIMENSION (5) :: STRG_LOC, STRG_THK
!
 COMMON /CONSTRUCTION/ SHIP_TYPE, TBHD_NUM, LBHD_NUM, TBHD_LOC,
 & LBHD_LOC, LBHD_THK, LBHD_MAT, LBHD_RUP, WEB_SPC,
 & DECK_THK, IBTM_THK, BTM_THK, DBL_HT, TDAB_THK,
 & STRG_NUM, STRG_WID, STRG_LOC, STRG_THK, TSTR_THK
!
!* Variables of properties of web frames (from deck to bottom)
!
! WEB_SUP - number of supports
! WEB_DEP - depth
! WEB_MAT - material grade at each support
! WEB_STF - stiffener spacing
! WEB_THK - thickness at each support
! WEB_SMZ - section modulus at each support
! WEB_SPN - unsupported span between each support
! WEB_GAP - supported length at each support
!
! SUP_MAT - material grade of each support
! SUP_ARA - axial area of each support
! SUP_GRA - gyration radius of each support
! SUP_LEN - critical length of each support
! STF_THK - smeared thickness of stiffeners
! STF_GRA - gyration radius of each stiffener w/ attached web plate
!
! LD_SPN - loading pattern on each span
! LD_BND - allowable bending load at each support
! LD_SHR - allowable shear load at each support
! LD_CMP - allowable compression load at each support
! LD_CRS - allowable cruching load per unit web span length
!
 INTEGER, DIMENSION (4) :: WEB_SUP
INTEGER, DIMENSION (4,4) :: WEB_MAT
REAL, DIMENSION (4) :: WEB_DEP, WEB_STF
REAL, DIMENSION (4,4) :: WEB_THK, WEB_SMZ
REAL, DIMENSION (4,3) :: WEB_SPN, WEB_GAP

INTEGER, DIMENSION (4,4) :: SUP_MAT
REAL, DIMENSION (4,4) :: SUP_ARA, SUP_GRA, SUP_LEN
REAL, DIMENSION (4) :: STF_THK, STF_GRA

REAL, DIMENSION (4,3,3) :: LD_SPN
REAL, DIMENSION (4,4) :: LD_BND, LD_SHR
REAL, DIMENSION (4,2:3) :: LD_CMP
REAL, DIMENSION (4) :: LD_CRS

COMMON /WEBS/ WEB_SUP, WEB_SPN, WEB_GAP, LD_SPN, LD_BND, LD_SHR,
&    LD_CMP, LD_CRS

SIGMA(GR,I) - yield & ultimate stresses and critical angle of steel (GR)
REAL, PARAMETER, DIMENSION (3,3) :: SIGMA=
& (/2.35E8, 3.15E8, 3.5E8,
& 4.2E8, 4.85E8, 5.25E8,
& 19.9, 17.3, 16.8/)

The following part contains the relevant functions
CONTAINS

!* Function LOAD is to calculate load pattern on each span
!* Output: LD_LN - load pattern (length): (1) & (3) w/o load, (2) w/load
!* FUNCTION LOAD() RESULT(LD_LN)
USE SHIP_DIM, ONLY : DEPTH1, DRAFT1, DRAFT2, BOW_HT
!
REAL, DIMENSION (4,3,3) :: LD_LN
!
K - number of side shell/bulkheads at one side (include centerline
bulkhead)
INTEGER :: I, J, K
!
REAL :: TOP, BTM, DDB, DDT
LOGICAL :: FT, FB
!
LD_LN=0.
K=LBHD_NUM/2
IF (K<LBHD_NUM*.5) K=K+1
!
DDB=DRAFT1-DRAFT2
DDT=BOW_HT-DRAFT2+DRAFT1
DO I=1, K
FB=.FALSE.
IF (DEPTH1<DDT) THEN
FT=.TRUE.
ELSE
FT=.FALSE.
END IF
TOP=DEPTH1
DO J=1, WEB_SUP(I)-1
TOP=TOP-WEB_GAP(I,J)
BTM=TOP-WEB_SPN(I,J)
IF (FT) THEN
LD_LN(I,J,1)=0.
ELSE IF (TOP<DDT) THEN
FT=.TRUE.
LD_LN(I,J,1)=0.
ELSE
    LD_LN(I,J,1)=MIN(WEB_SPN(I,J), TOP-DDT)
END IF
IF (FB) THEN
    LD_LN(I,J,3)=WEB_SPN(I,J)
ELSE IF (BTM<DDB) THEN
    FB=.TRUE.
    LD_LN(I,J,3)=MIN(WEB_SPN(I,J), DDB-BTM)
ELSE
    LD_LN(I,J,3)=0.
END IF
LD_LN(I,J,2)=WEB_SPN(I,J)-LD_LN(I,J,1)-LD_LN(I,J,3)
TOP=BTM
END DO
END DO
END FUNCTION LOAD

! * Function A_BND is to calculate allowable bending load at each support
! Output: BND - allowable bending moments
!
FUNCTION A_BND() RESULT(BND)
!
REAL, DIMENSION (4,4) :: BND
!
! K - number of side shell/bulkheads at one side (include centerline
bulkhead)
INTEGER :: I, J, K
!
BND=0.
K=LBHD_NUM/2
IF (K<LBHD_NUM*.5) K=K+1
DO I=1, K
    DO J=1, WEB_SUP(I)
        BND(I,J)=WEB_SMZ(I,J)*SIGMA(WEB_MAT(I,J),1)*1.12
    END DO
END DO
END FUNCTION A_BND

! * Function A_SHR is to calculate allowable shear load at each support
! Output: SHR - allowable shear forces
!
FUNCTION A_SHR() RESULT(SHR)
USE GENERAL, ONLY : PI, EM, PR
!
REAL, DIMENSION (4,4) :: SHR
!
! K - number of side shell/bulkheads at one side (include centerline
bulkhead)
INTEGER :: I, J, K
!
! TOU_C - critical shear stress
! TOU_Y - yielding shear stress
! SIGMA_TY - field tensile stress
REAL :: TOU_C, TOU_Y, SIGMA_TY, DOA, DD, THETA
!
SHR=0.
K=LBHD_NUM/2
IF (K<LBHD_NUM*.5) K=K+1
DO I=1, K
    IF (WEB_DEP(I)<WEB_STF(I)) THEN
        DOA=WEB_DEP(I)/WEB_STF(I)
        DD=WEB_DEP(I)
        ELSE
DOA=WEB_STF(I)/WEB_DEP(I)
DD=WEB_STF(I)
END IF
DO J=1, WEB_SUP(I)
  TOU_C=(5.34+4.*DOA*DOA)*PI*PI*EM*WEB_THK(I,J)*WEB_THK(I,J)
   /(12..*(1-PR*PR)*DD*DD)
  TOU_Y=.58*SIGMA(WEB_MAT(I,J),1)
  IF (TOU_Y<TOU_C) THEN
    SHR(I,J)=TOU_Y*WEB_DEP(I)*WEB_THK(I,J)
  ELSE
    SIGMA_TY=SIGMA(WEB_MAT(I,J),1)*(1.-TOU_C/TOU_Y)
    THETA=ATAN(DOA)*.5
    SHR(I,J)=TOU_C*WEB_DEP(I)*WEB_THK(I,J)
    & *SIGMA_TY*SIN(THETA)*COS(THETA)*WEB_THK(I,J)
    & *(WEB_DEP(I)-WEB_STF(I)*TAN(THETA))
  END IF
END DO
END DO
END FUNCTION A_SHR

!* Function A_CMP is to calculate allowable compression load at each support
!* Output: CMP - allowable compression load
!*  
!  FUNCTION A_CMP() RESULT(CMP)
!  USE GENERAL, ONLY : PI, EM
!  
!  REAL, DIMENSION (4,2:3) :: CMP
!  !  K - number of side shell/bulkheads at one side (include centerline bulkhead)
!  INTEGER :: I, J, K
!
!  CMP=0.
!  K=LBHD_NUM/2
!  IF (K<LBHD_NUM*.5) K=K+1
!  DO I=1, K
!     DO J=2, WEB_SUP(I)
!       CMP(I,J)=SUP_ARA(I,J)*SIGMA(SUP_MAT(I,J),1)
!       & *(1.-SIGMA(SUP_MAT(I,J),1)/(4.*PI*PI*EM)
!       & *(SUP_LEN(I,J)/SUP_GRA(I,J))**2)
!     END DO
!   END DO
END FUNCTION A_CMP

!* Function A_CRS is to calculate allowable crushing load at each support
!* Output: CRS - allowable crushing load per unit web span length
!*  
!  FUNCTION A_CRS() RESULT(CRS)
!  USE GENERAL, ONLY : PI, EM
!
!  REAL, DIMENSION (4) :: CRS
!  !  K - number of side shell/bulkheads at one side (include centerline bulkhead)
!  INTEGER :: I, J, K
!
!  CRS=0.
!  K=LBHD_NUM/2
!  IF (K<LBHD_NUM*.5) K=K+1
DO I=1, K
    MIN_THK=WEB_THK(I, 1)
    DO J=2, WEB_SUP(I)
        IF (MIN_THK>WEB_THK(I, J)) MIN_THK=WEB_THK(I, J)
    END DO
    IF (I==2.AND.SHIP_TYPE==2) THEN
        COEF=1.
    ELSE
        COEF=(1.-SIGMA(1,1)*(WEB_DEP(I)/STF_GRA(I))**2
        &/(4.*PI*PI*EM))
    END IF
    CRS(I)=(STF_THK(I)+MIN_THK)*SIGMA(1,1)*COEF
END DO
END FUNCTION A_CRS

!* Function BEND is to calculate bending moment at each support
!* Input : K - current longitudinal bulkhead from side
!* S - current web span from top
!* FRC - applied force on web per unit web length
!* Output: BND - bending moment at each support
!* 
FUNCTION BEND(K, S, FRC) RESULT(BND)

INTEGER, INTENT (IN) :: K, S
REAL, DIMENSION (2), INTENT (IN) :: FRC
REAL, DIMENSION (2,2) :: BND

REAL :: A
REAL, DIMENSION (2) :: Q

BND(1,:)=FRC*WEB_SPN(K,S)**2/12.
BND(2,:)=FRC*WEB_SPN(K,S)**2/12.
IF (LD_SPN(K,S,1)>0.) THEN
    A=LD_SPN(K,S,1)/WEB_SPN(K,S)
    Q=FRC*LD_SPN(K,S,1)**2/12.
    BND(1,:)=BND(1,:)-Q*(6.-8.*A+3.*A*A)
    BND(2,:)=BND(2,:)-Q*(4.-3.*A)*A
END IF
IF (LD_SPN(K,S,3)>0.) THEN
    A=LD_SPN(K,S,3)/WEB_SPN(K,S)
    Q=FRC*LD_SPN(K,S,3)**2/12.
    BND(1,:)=BND(1,:)-Q*(4.-3.*A)*A
    BND(2,:)=BND(2,:)-Q*(6.-8.*A+3.*A*A)
END IF
END FUNCTION BEND

!* Function SHRG is to calculate shearing force at each support
!* Input : K - current longitudinal bulkhead from side
!* S - current web span from top
!* FRC - applied force on web per unit web length
!* Output: SHR - shearing force at each support
!* 
FUNCTION SHRG(K, S, FRC) RESULT(SHR)

INTEGER, INTENT (IN) :: K, S
REAL, DIMENSION (2), INTENT (IN) :: FRC
REAL, DIMENSION (2,2) :: SHR

REAL :: A
REAL, DIMENSION (2) :: Q

SHR(1,:)=FRC*WEB_SPN(K,S)/2.
SHR(2,:)=FRC*WEB_SPN(K,S)/2.
IF (LD_SPN(K,S,1)>0.) THEN
  A=LD_SPN(K,S,1)/WEB_SPN(K,S)
  Q=FRC*LD_SPN(K,S,1)/2.
  SHR(1,:)=SHR(1,:)-Q*(2.-2.*A*A+A*A*A)
  SHR(2,:)=SHR(2,:)-Q*(2.-A)*A*A
END IF
IF (LD_SPN(K,S,3)>0.) THEN
  A=LD_SPN(K,S,3)/WEB_SPN(K,S)
  Q=FRC*LD_SPN(K,S,3)/2.
  SHR(1,:)=SHR(1,:)-Q*(2.-A)*A*A
  SHR(2,:)=SHR(2,:)-Q*(2.-2.*A*A+A*A*A)
END IF
END FUNCTION SHRG

!* Function LD_FCT is to calculate actual loads vs allowable loads of webs
!* Input : LBHD_CUR - current longitudinal bulkhead
!*   FRC_APL - applied force on web per unit web length
!* Output: FCT - maximum load factor
!*   STAT - governing mode: 1 - crushing of web itself, 0 - otherwise
!* FUNCTION LD_FCT(LBHD_CUR, FRC_APL, STAT) RESULT (FCT)
INTEGER, INTENT (IN) :: LBHD_CUR
REAL, DIMENSION (2), INTENT (IN) :: FRC_APL
REAL, DIMENSION (2) :: FCT
INTEGER, INTENT (OUT) :: STAT
!
! K - current number of bulkhead counted from closer ship side
INTEGER :: I, J, K
!
! DH2 - flag showing current bulkhead is the 2nd of double hull
LOGICAL :: DH2
!
! ACT_LD - actual load on web
! ACT_CMP - actual compression load
REAL, DIMENSION (2,2) :: ACT_LD
REAL, DIMENSION (2) :: FCT_TMP
REAL, DIMENSION (2:3,2) :: ACT_CMP
!
K=LBHD_NUM/2
IF (K<LBHD_NUM*.5) K=K+1
IF (LBHD_CUR<=K) THEN
  K=LBHD_CUR
ELSE
  K=K*2-LBHD_CUR
END IF
!
IF (SHIP_TYPE==2.AND.(LBHD_CUR==2.OR.LBHD_CUR==LBHD_NUM)) THEN
  DH2=.TRUE.
ELSE
  DH2=.FALSE.
END IF
!
FCT=0.
STAT=0
ACT_CMP=0.
DO I=1, WEB_SUP(K)-1
  ! bending moment
  ACT_LD=BEND(K, I, FRC_APL)
  FCT_TMP (1)=MIN(ACT_LD (1,1)/LD_BND(K,I),
  & ACT_LD (2,1)/LD_BND(K,I+1))
  FCT_TMP (2)=MIN(ACT_LD (1,2)/LD_BND(K,I),
  & ACT_LD (2,2)/LD_BND(K,I+1))
  IF (FCT (1)<FCT_TMP (1)) FCT (1)=FCT_TMP (1)
IF (FCT(2)<FCT_TMP(2)) FCT(2)=FCT_TMP(2)

! shear force
ACT_LD=SHRG(K, I, FRC_APL)
FCT_TMP(1)=MIN(ACT_LD(1,1)/LD_SHR(K,I),
& ACT_LD(2,1)/LD_SHR(K,I+1))
FCT_TMP(2)=MIN(ACT_LD(1,2)/LD_SHR(K,I),
& ACT_LD(2,2)/LD_SHR(K,I+1))
IF (FCT(1)<FCT_TMP(1)) FCT(1)=FCT_TMP(1)
IF (FCT(2)<FCT_TMP(2)) FCT(2)=FCT_TMP(2)

! compression
IF (WEB_SUP(K)>2) THEN
  IF (I>1) ACT_CMP(I,:)=ACT_CMP(I,:)+ACT_LD(1,:)
  IF (I<WEB_SUP(K)-1)
    & ACT_CMP(I+1,:)=ACT_CMP(I+1,:)+ACT_LD(2,:)
END IF
END DO

IF (WEB_SUP(K)>2) THEN
  DO I=2, WEB_SUP(K)-1
    FCT_TMP=ACT_CMP(I,:)/LD_CMP(K,I)
    IF (FCT(1)<FCT_TMP(1)) FCT(1)=FCT_TMP(1)
    IF (FCT(2)<FCT_TMP(2)) FCT(2)=FCT_TMP(2)
  END DO
END IF

! crushing
IF (.NOT.DH2) THEN
  FCT_TMP=FRC_APL/LD_CRS(K)
  IF (FCT(1)<FCT_TMP(1)) THEN
    FCT(1)=FCT_TMP(1)
    STAT=1
  END IF
  IF (FCT(2)<FCT_TMP(2)) THEN
    FCT(2)=FCT_TMP(2)
    STAT=1
  END IF
END IF

END FUNCTION LD_FCT

END MODULE SHIP_STR

A.3.3 collision_mod.for

!*** COLLISION_MOD.FOR by Donghui Chen
!** VERSION: 2.1 DATE: December 10, 1999
!** REVISION NOTES:
! v2.1
! - Incorporate definite bow depth
! ! v2.0
! - Incorporate double hull features
! - MEMBRANE EFFECTS
! - Consider friction
! - Consider force to propogate yielding point
! - MINORSKY'S MECHANISM
! - Consider side shell & longitudinal bulkheads
- Consider stringers

v1.2
- MEMBRANE EFFECTS
  - Using Rosenblatt's method (supporting web could be damaged)

v1.1
- Separate out from MODULE.FOR.
- MEMBRANE EFFECTS
  - Using Rosenblatt's method (supporting web undamaged)

v1.0
- MEMBRANE EFFECTS
  - Let upper level procedure determine the effective breadth, for which
    the depth of struck ship is suggested.
  - It is suggested to use smeared plating thickness.
  - Revised the formula used to calculate the membrane forces.
- MINORSKY MECHANISM
  - Revised method to calculate damaged area.
- VIRTUAL MASS
  - Correct the formula (Rev 02/17)
  - Add one variable for determinant (Rev 03/04)

** CONTENTS:
- MODULES
  SHIP_COL - defines variables of collision scenario
  COL_FORCE - defines functions for collision forces

** Module SHIP_COL defines variables of collision scenario and function used
  calculate time step of simulation

CONTENTS:
- COMMON BLOCKS
  STRUCK_COL
  STRIKE_COL
- FUNCTION
  TIME_STEP - calculates the time step used in time domain simulation

MODULE SHIP_COL

!* Variables of collision scenario for struck ship

X1 - the location of the ship in global coordinates x and y
V1 - the speed of the ship on global x and y axis
OMEGA1 - the angle between x-axis and longitudinal direction of ship
W1 - the angular speed of the ship
LOC - the location of the impact point along the ship length
    (measured from midship)
PEN - the depth of penetration across the ship (transversely)
REAL, DIMENSION (2) :: X1, V1
REAL :: OMEGA1, W1, LOC, PEN

COMMON /STRUCK_COL/ X1, V1, OMEGA1, LOC, PEN

!* Variables of collision scenario for striking ship

X2 - the location of the ship in global coordinates x and y
V2 - the speed of the ship on global x and y axis
OMEGA2 - the angle between x-axis and longitudinal direction of ship
W2 - the angular speed of the ship

REAL, DIMENSION (2) :: X2, V2
REAL :: OMEGA2, W2

COMMON /STRIKE_COL/ X2, V2, OMEGA2, W2

The following part contains the relevant functions

CONTAINS

!* Function TIME_STEP is to calculate the time step used in time domain
! collision analysis (i.a.w. B. L. Hutchison's method, 1/200 of duration)
! Input : BEAM, DRAFT, DISP, LBP
! THK - total plating thickness of decks and bottom shells
! HEA - half-entrance angle of striking ship
! MV1 - virtual mass (m11) of struck ship
! MV2 - virtual mass (m11) of striking ship
! Output: TIME_STEP - time step for time domain simulation
!
REAL FUNCTION TIME_STEP(THK, HEA, MV1, MV2)
REAL, INTENT (IN) :: THK, HEA
REAL :: MV1, MV2
TIME_STEP=SQRT(MV1*MV2/(THK*TAND(HEA)*(MV1+MV2)))*8.09216E-8
END FUNCTION TIME_STEP

END MODULE SHIP_COL

** Module COL_FORCE defines functions used for calculating reaction forces

CONTENTS:
- VARIABLES
- FUNCTIONS
  ACC - calculates the acceleration of ships (Rev 03/04)
  EFF_DP - calculate effect depth of striking bow
  EFF_BR - calculate effect breadth of membrane tension
  DEFL_LIM - calculates the limitation of deflection
  EL - calculates elongation of each segment
  REL_MOVE - calculate relative movement of striking bow
  PSHWEB - calculate the deformation of webs by striking bow
  PEN_BOW - determines the points coordinates of penetrated bow
-SUBROUTINES
  MEMB_FORCE - calculates the reaction force by membrane effect
  MINO_DECK - calculates Minorsky's force in decks
  MINO_SIDE - calculates Minorsky's force in bulkheads
  MINO_FORCE - calculates the reaction force by Minorsky mechanism

MODULE COL_FORCE

!* Variables used for calculations of reaction forces
! E1 - strain on L1, E1 = .1
! DKE - Minorsky's coefficient
! FRO - factor to determine right angle or oblique collision
REAL, PARAMETER :: E1=.1, DKE=47.1E6, FRO=.02
!
! LBHD_CUR - set the current longitudinal bulkhead being or to be reached
! REACHED - flag showing one or two longitudinal bulkheads being reached
INTEGER :: LBHD_CUR, REACHED
!
! LOC0 - initial impact location on longitudinal bulkheads
! D_LOC0 - initial shift of impact location
! MEMB_EN0 - membrane energy absorbed till previous time step
! MINO_EN0 - total absorbed Minorsky's energy (for debug purpose only)
DW0 - web deformation due to direct impact
OMEGAR - OMEGA1-OMEGA2
FL_MAX - limitation of longitudinal force

REAL, DIMENSION (2) :: LOC0, MEMB_EN0, DW0, OMEGAR
REAL :: D_LOC0, MINO_EN0, FL_MAX, DF2

REAL :: INNER, OUTER

BP1 - nodes of penetrating bow before this time step
BP2 - nodes of penetrating bow after this time step
REAL, DIMENSION (5,2) :: BP1, BP2

The following part contains the relevant functions
CONTAINS

!* Function ACC is to calculate the accelerations of ships (Rev 03/04)
! Input : FORCE - the applied force
!   ZETA - force direction in global system
!   MV - virtual mass of the ships
! Output: VAC - acceleration in global system
!
FUNCTION ACC(FORCE, ZETA, MV) RESULT(VAC)
!
IMPLICIT NONE
REAL, INTENT (IN) :: FORCE, ZETA
REAL, INTENT (IN), DIMENSION (4) :: MV
REAL, DIMENSION (2) :: VAC

REAL :: FX, FY

FX=FORCE*COSD(ZETA)
FY=FORCE*SIND(ZETA)

IF (MV(4)<>0.) THEN
  VAC(1)=(FX*MV(2)-FY*MV(3))/MV(4)
  VAC(2)=(FY*MV(1)-FX*MV(3))/MV(4)
ELSE
  VAC=0.
END IF
END FUNCTION ACC

!* Function EFF_DP is to calculate effect depth of striking bow
! Input : SHIP_DIM
! Output: EFF_DP - effective depth of striking bow
!
REAL FUNCTION EFF_DP()
USE SHIP_DIM, ONLY : DEPTH1, DRAFT1, DRAFT2, BOW_HT
!
IMPLICIT NONE
!
REAL :: DDT, DDB
!
DDT=BOW_HT-DRAFT2+DRAFT1
DDB=DRAFT1-DRAFT2
!
EFF_DP=MIN(DEPTH1, DDT)-MAX(0., DDB)
!
END FUNCTION EFF_DP

!* Function EFF_BR is to calculate effect breadth of membrane tension
! Input : K - current bulkhead number from side
!   RUP - flag of rupture
!   SHIP_DIM, SHIP_STR
! Output: EBR - effective breadth of membrane effect
!
(1): bow top to support; (2): striking depth;
FUNCTION EFF_BR(K, RUP) RESULT(EBR)
USE SHIP_DIM, ONLY : DEPTH1, DRAFT1, DRAFT2, BOW_HT
USE SHIP_STR, ONLY : SHIP_TYPE, DBL_HT, STRG_NUM, STRG_LOC
IMPLICIT NONE
INTEGER, INTENT (IN) :: K, RUP
REAL, DIMENSION (4) :: EBR
INTEGER :: I
REAL :: DDT, DDB
LOGICAL :: DH2

DDT=BOW_HT-DRAFT2+DRAFT1
DDB=DRAFT1-DRAFT2
IF (SHIP_TYPE==2.AND.K==2) THEN
DH2=.TRUE.
ELSE
DH2=.FALSE.
END IF

EBR(2)=MIN(DEPTH1, DDT)-MAX(0., DDB)
IF (DEPTH1>DDT) THEN
EBR(1)=DEPTH1-DDT
IF (K==1.OR.DH2) THEN
I=1
DO WHILE (STRG_LOC(I)>DDT)
EBR(1)=STRG_LOC(I)-DDT
I=I+1
END DO
ELSE
EBR(1)=0.
END IF
ELSE
EBR(1)=0.
END IF

IF (DDB>0.) THEN
EBR(3)=DDB
IF (SHIP_TYPE==2.AND.DDB>DBL_HT) EBR(3)=DDB-DBL_HT
IF (K==1.OR.DH2) THEN
I=STRG_NUM
DO WHILE (DDB>STRG_LOC(I))
EBR(2)=DDB-STRG_LOC(I)
I=I-1
END DO
ELSE
EBR(3)=0.
END IF
ELSE
EBR(3)=0.
END IF

IF (RUP>1) THEN
EBR(4)=0.
ELSE IF (RUP==1) THEN
EBR(4)=EBR(2)
ELSE
EBR(4)=EBR(2)+(EBR(1)+EBR(3))*0.5
END IF
END FUNCTION EFF_BR

!* Function DEFL_LIM is to calculate the limitation of deflection
Input : ND - collision angle: 0 - right angle; 1 - forward; 2 - aftward
NC - number of crushed webs: 1 - forward; 2 - aft
LW - distances: 1 - to forward web; 2 - to aft web
WEB_SPC - web spacing
THETA1 - critical bending angle
THETA2 - actual bending angle at webs: 1 - forward; 2 aft
Output: DEFL_LIM - limitation of deflection

REAL FUNCTION DEFL_LIM(ND, NC, LW, WEB_SPC, THETA1, THETA2)

IMPLICIT NONE
INTEGER, INTENT (IN) :: ND
INTEGER, DIMENSION (2), INTENT (IN) :: NC
REAL, DIMENSION (2), INTENT (IN) :: LW
REAL, INTENT (IN) :: WEB_SPC, THETA1
REAL, DIMENSION (2), INTENT (IN) :: THETA2

INTEGER :: I, J
REAL, PARAMETER :: PHI=24.62 ! angle based on 0.1 strain rate
REAL :: LWS, DEFL_TMP, THETA, THETAC, THETAT, DELTA, L1

IF (ND>0) THEN
    THETA=2.*THETA1
    THETAC=MIN(THETA, THETA2(ND))
    IF (THETA+THETAC*NC(ND)>90.) THEN
        DEFL_LIM=10000.
    ELSE
        DEFL_LIM=LW(ND)*TAND(THETA+THETAC*NC(ND))
        DO I=1, NC(ND)
            DEFL_LIM=DEFL_LIM+WEB_SPC*TAND(THETA+THETAC*(I-1))
        END DO
    END IF
    L1=LW(3-ND)
    LWS=LW(1)+LW(2)
    THETAT=ATAND(SQRT((8400.*LWS*LWS-7560.*LWS*L1-399.*L1*L1)
        & 
        /(40000.*LWS*LWS+8000.*LWS*L1+400.*L1*L1)))
ELSE
    DEFL_LIM=10000.
    THETAT=PHI
END IF

DO I=1, 2
    IF (I<>ND) THEN
        DEFL_TMP=LW(I)*TAND(THETAT)
        THETAC=THETA2(I)
        DO J=1, NC(I), 1
            DEFL_TMP=DEFL_TMP+WEB_SPC*TAND(MAX(0., THETAT-THETAC*J))
        END DO
        DEFL_LIM=MIN(DEFL_TMP, DEFL_LIM)
    END IF
END DO

END FUNCTION DEFL_LIM

!* Function EL is to calculate the elongation of each segment
! Input : DF1 - deflection at the first node
! DF2 - deflection at the second node
! L - length of the segment
! Output: EL - elongation of the segment
!
REAL FUNCTION EL(DF1, DF2, L)

IMPLICIT NONE
REAL, INTENT (IN) :: DF1, DF2, L
REAL :: D_DF

D_DF=DF1-DF2
EL=SQRT(D_DF*D_DF+L*L)-L

END FUNCTION EL

!* Function PSHWEB is to calculate the deformation of webs by striking bow
!* Input : DN - side of striking:  1 - forward; 2 - aft
!*   NW - number of web from impact
!*   NL - current longitudinal bulkhead
!*   LW - distances from impact: 1 - to forward web; 2 - to aft web
!*   SWB - web spacing
!* Output: PSHWEB - pushing distance by striking bow
*
REAL FUNCTION PSHWEB(DN, NW, NL, LW, SWB)
USE GENERAL
USE SHIP_STR
USE SHIP_DIM

IMPLICIT NONE
INTEGER, INTENT (IN) :: DN, NW, NL
REAL, DIMENSION (2), INTENT (IN) :: LW
REAL, INTENT (IN) :: SWB

INTEGER :: K
REAL :: LL, LC
REAL, DIMENSION (5,2) :: BP

K=LBHD_NUM/2
IF (K<LBHD_NUM*.5) K=K+1
IF (NL<=K) THEN
  K=NL
ELSE
  K=K*2-NL
END IF

LL=LW(DN)+SWB*(NW-1)
PSHWEB=0.
LC=BEAM1*.5-LBHD_LOC(NL)
BP=BP2
IF (BP(3,1)>LC.AND.(SHIP_TYPE<>2. OR.NL<>2)) THEN
  IF (BP2(2,2)>LC) THEN
    BP(1,:)=INTCPTP(BP2(1,:), BP2(2,:), LC)
  ELSE IF (BP2(3,2)>LC) THEN
    BP(1,:)=INTCPTP(BP2(2,:), BP2(3,:), LC)
    BP(2,:)=BP(1,:)
  END IF
  IF (BP2(4,2)>LC) THEN
    BP(5,:)=INTCPTP(BP2(4,:), BP2(5,:), LC)
  ELSE IF (BP2(3,2)>LC) THEN
    BP(4,:)=INTCPTP(BP2(3,:), BP2(4,:), LC)
    BP(5,:)=BP(4,:)
  END IF
END IF

SELECT CASE (DN)
CASE (1)
  IF (BP(3,1)-BP(2,1)>=LL) THEN
    PSHWEB=BP(3,2)-(BP(3,2)-BP(2,2))*LL/(BP(3,1)-BP(2,1))
    & -LC
  ELSE IF (BP(3,1)-BP(1,1)>=LL) THEN
    PSHWEB=BP(2,2)*(1.-((LL-(BP(3,1)-BP(2,1))))
END IF
& (BP(2,1)-BP(1,1))=LC

END IF
CASE (2)
IF (BP(4,1)-BP(3,1)>=LL) THEN
  PSHWEB=BP(3,2)-(BP(3,2)-BP(4,2))*LL/(BP(4,1)-BP(3,1))
  & -LC
ELSE IF (BP(5,1)-BP(3,1)>=LL) THEN
  PSHWEB=BP(2,2)*(1.-((BP(4,1)-BP(3,1))
  & /(BP(5,1)-BP(4,1)))-LC
END IF
END SELECT
END IF
!
END FUNCTION PSHWEB
!
!* Subroutine MEMB_FORCE is to calculate the reaction force caused by
! membrane effects
! Input : LOC - the location of the impact point along the ship length
! (measured from FP)
! D_LOC - shift of impact point along the length in each time step
! D_PEN - increment of penetration during each time step
! DEFL - deflection of the side shell/bulkheads at impact point
! M - total number of scenarios (for testing output)
! SHIP_STR module
! Output: FORCE - reaction force caused by membrane effects
! ANGLE - striking force angle in LOC-PEN system
! MOMNT - reaction moment about the origin of LOC-PEN system
! DEFL - deflection (updated when ruptured)
!
SUBROUTINE MEMB_FORCE(LOC, D_LOC, D_PEN, DEFL,
& FORCE, ANGLE, MOMNT, M)
USE GENERAL, ONLY : V_ABS, PI
USE SHIP_STR
USE SHIP_DIM
!
IMPLICIT NONE
INTEGER, OPTIONAL, INTENT (IN) :: M
REAL, INTENT (IN) :: LOC, D_LOC, D_PEN
REAL, DIMENSION (2), INTENT (INOUT) :: DEFL
REAL, INTENT (OUT) :: FORCE, ANGLE, MOMNT
!
! RFP - ratio of propagation force to yielding x area of bulkheads
! CFR - friction coefficient
REAL, PARAMETER :: RFP=.025, CFR=.15
!
! DH - flag showing current bulkhead is the 1st of double hull
! DH2 - flag showing current bulkhead is the 2nd of double hull
LOGICAL :: DH, DH2
!
! CR - reached longitudinal bulkhead from 1st one reached at the same time
! K - reached bulkhead counted from closer ship side
! K2 - 2nd skin of double hull counted from closer ship side
! NLC - longitudinal bulkhead under consideration from side shell
! N1 - side ahead of strike: 1 - forward; 2 - aft
! N10 - side ahead of initial strike: 1 - forward; 2 - aft
! NC - number of crushed webs: 1 - forward; 2 - aft
! NWB - total web spacing between bulkheads
! STAT - governing mode: 1 - crushing of web itself, 0 - otherwise
INTEGER :: I, J, CR, K, K2, NLC, N1, N10, NN, NWB, STAT
INTEGER, DIMENSION (2) :: NC=0
!
! BR - (1): bow top to support; (2): striking depth;
! (3): bow bottom to support; (4): effective breadth
! WA - maximum deflection
! ET - total membrane elongation: 1 - forward; 2 - aft
! ET2 - total membrane elongation of 2nd skin of double hull
! BAV - vertical bending angle at top/bottom
! BA2 - bending angle at 2nd skin of double hull
! DEFLA - effective deflection after the time step
! DEFLA2 - effective deflection of 2nd skin after the time step
! MISC_EN - energy absorbed to propagate yielding point
! LOST_EN - energy absorbed during the time step
! WEB_EN - energy absorbed by webs till this time step
REAL, DIMENSION (4) :: BR
REAL :: WA, ET2, BAV, BA2, DEFLA, DEFLA2
REAL :: MISC_EN, LOST_EN, WEB_EN
REAL, DIMENSION (2) :: ET, WEB_EN
!
! LWS - distances between flank webs
! TN - normal membrane tension force per unit width
! TN2 - normal membrane tension force per unit width in 2nd skin
! PWFM - max force per width on webs
! DWFT - deflection at current web frame
! PSHT - deformation of web caused by striking bow
! LR - impact location relative to transverse bulkhead
! LRO - initial impact location relative to transverse bulkhead
! LWO - initial impact location relative to the closest web
REAL :: LWS, TN, TN2, PWFM, DWFT, PSHT, PCS, LR, LRO, LWO
REAL :: F1, F2, FW, FWC, B2S, MOMNT1, MOMNT2
!
! MEMB_EN - membrane energy absorbed till this time step
! LW - distances: 1 - to forward web; 2 - to aft web
! T - membrane tension force per width: 1 - forward; 2 - aft
! PWF - force per width on webs: 1 - forward; 2 - aft
! RM - load factor: 1 - forward; 2 - aft
! THETA - bending angle on webs: 1 - forward; 2 - aft
! DWF - deflection at web locations: 1 - forward; 2 - aft
! FORC - total force components in LOC-PEN system
REAL, DIMENSION (2) :: MEMB_EN, LW, T, PWF, RM, THETA, FORC
REAL, DIMENSION (2,15) :: DWF
!
MEMB_EN=0.
WEB_EN=0.
MOMNT1=0.
MOMNT2=0.
FORC=0.
ANGLE=90.
DH=.FALSE.
DH2=.FALSE.
IF (SHIP_TYPE==2) THEN 
  IF (LBHD_CUR==1.OR.LBHD_CUR==LBHD_NUM-1) THEN 
    IF (LBHD_RUP(LBHD_CUR+1)>1) THEN 
      DH=.FALSE.
    ELSE 
      DH=.TRUE.
    END IF 
  END IF 
  IF (LBHD_CUR==2.OR.LBHD_CUR==LBHD_NUM) DH2=.TRUE.
END IF 
!
ruptured structure
WEBEC_EN=0.
FWC=0.
BR(2)=EFF_DP()
DO NLC=1, LBHD_CUR-1 
  B2S=BEAM1*.5-LBHD_LOC(NLC)
\[ K = \text{LBHD\_NUM}/2 \]
IF (K<\text{LBHD\_NUM}*.5) K=K+1
IF (NLC<=K) THEN
K=NLC
ELSE
K=K*2-NLC
END IF
\[ TN = .5*\text{LBHD\_THK}(K) \times (\text{SIGMA} (\text{LBHD\_MAT}(K),1)+\text{SIGMA} (\text{LBHD\_MAT}(K),2)) \]
\[ \text{PWF} = \frac{TN}{\text{WEB\_SPC}} \]
\[ \text{RM} = \text{LD\_FCT} (\text{NLC}, \text{PWF}, \text{STAT}) \]
\[ \text{PWF}\_M = \frac{\text{PWF}(1)}{\text{RM}(1)} \]

! To determine the impact location relative to webs
\[ \text{LW}(1) = \text{MOD} (\text{LOC}-\text{TBHD\_LOC}(1), \text{WEB\_SPC}) \]
\[ \text{LW}(2) = \text{LWS}-\text{LW}(1) \]
\[ \text{NC}(1) = \left( \frac{\text{LOC}-\text{TBHD\_LOC}(1)}{\text{WEB\_SPC}} \right) \]
\[ \text{NC}(2) = \left( \frac{\text{TBHD\_LOC} (\text{TBHD\_NUM}) - \text{LOC}}{\text{WEB\_SPC}} \right) \]

DO I=1, 2
DO J=1, NC(I)
\[ \text{PSHT} = \text{PSHWEB}(I, J, \text{NLC}, \text{LW}, \text{WEB\_SPC}) \]
IF (PSHT<=0.) EXIT
IF (PSHT<=\text{WEB\_DEP}(K)) THEN
\[ \text{FWC} = \text{FWC} + \text{PWF}\_M \times \text{BR}(2) \]
\[ \text{MOMNT}1 = \text{MOMNT}1 - \text{PWF}\_M \times \text{BR}(2) \]
& \* \text{SIGN} (\text{CFR}, \text{D\_LOC}) \* (\text{PSHT}+\text{B2S})
\[ \text{MOMNT}2 = \text{MOMNT}2 + \text{PWF}\_M \times \text{BR}(2) \times (\text{LOC}+\text{SIGN} (\text{LW}(I)+(J-1)) \]
& \* \text{WEB\_SPC}, I-1.5))
END IF
\[ \text{WEBC\_EN} = \text{WEBC\_EN} + \text{PWF}\_M \times \text{PSHT} \]
END DO
END DO
WEBC\_EN=0.
FWC=0.
MOMNT1=0.
MOMNT2=0.

! reached structure
F2=0.
FW=0.
DO CR=1, \text{REACHED}
\[ \text{NLC} = \text{LBHD\_CUR} + \text{CR} - 1 \]
\[ \text{B2S} = \text{BEAM}1\times .5-\text{LBHD\_LOC}(\text{NLC}) \]
\[ K = \text{LBHD\_NUM}/2 \]
IF (K<\text{LBHD\_NUM}*.5) K=K+1
IF (NLC<=K) THEN
K=NLC
ELSE
K=K*2-NLC
END IF
BR=\text{EFF\_BR}(K, \text{LBHD\_RUP}(\text{NLC}))

IF (\text{LBHD\_RUP}(\text{NLC})<1.AND.\text{BR}(1)>0.) THEN
\[ \text{BAV} = .5\times \text{ATAND} (\text{DEFL}(\text{CR})/\text{BR}(1)) \]
IF (\text{BAV}<\text{SIGMA} (\text{LBHD\_MAT}(K),3)) THEN
\[ \text{LBHD\_RUP}(\text{NLC}) = 1 \]
\[ \text{BR}(4) = \text{BR}(2) \]
END IF
END IF
END IF
IF (\text{LBHD\_RUP}(\text{NLC})<1.AND.\text{BR}(3)>0.) THEN
\[ \text{BAV} = .5\times \text{ATAND} (\text{DEFL}(\text{CR})/\text{BR}(3)) \]
IF (\text{BAV}<\text{SIGMA} (\text{LBHD\_MAT}(K),3)) THEN

132
LBHD_RUP(NLC)=1
BR(4)=BR(2)
END IF
END IF

! N1=0
N10=0
IF (LOC<LOC0(CR)) N10=1
IF (LOC>LOC0(CR)) N10=2
!

! Average membrane tension force
TN=.5*LBHD_THK(K)*(SIGMA(LBHD_MAT(K),1)+SIGMA(LBHD_MAT(K),2))
T=TN
IF (DH) THEN
IF (LBHD_CUR==1) K2=2
IF (LBHD_CUR==LBHD_NUM-1) K2=1
TN2=.5*LBHD_THK(K2)
&   *(SIGMA(LBHD_MAT(K2),1)+SIGMA(LBHD_MAT(K2),2))
END IF
!

! To determine the impact relative to transverse bulkheads
DO I=2, TBHD_NUM
LR=LOC-TBHD_LOC(I)
LR0=LOC0(CR)-TBHD_LOC(I)
IF (ABS(LR)<.1) THEN
   NWB=INT((TBHD_LOC(I+1)-TBHD_LOC(I-1))/WEB_SPC)
   NC(1)=INT((TBHD_LOC(I)-TBHD_LOC(I-1))/WEB_SPC)-1
   NC(2)=INT((TBHD_LOC(I+1)-TBHD_LOC(I))/WEB_SPC)-1
   IF (ABS(LR0)<.1) THEN
      IF (LR*LR0<0.) THEN
         LBHD_RUP(NLC)=2
         IF (NLC==1) THEN
            OUTER=DEFL(1)
         END IF
         IF (NLC==2) THEN
            INNER=DEFL(1)+B2S
            DF2=0.
         END IF
      END IF
      EXIT
   END IF
END IF
ELSE
IF (LR<0.) THEN
   NWB=INT((TBHD_LOC(1)-TBHD_LOC(I-1))/WEB_SPC)
   NC(1)=INT((LOC-TBHD_LOC(I-1))/WEB_SPC)
   NC(2)=INT((TBHD_LOC(I)-LOC)/WEB_SPC)
   LW0=MOD(LOC0(CR)-TBHD_LOC(1), WEB_SPC)
   LW0=MIN(LW0, WEB_SPC-LW0)
   IF (LW0>.1) THEN
      IF (INT(-LR0/WEB_SPC)<>NC(2)) THEN
         LBHD_RUP(NLC)=2
         IF (NLC==1) THEN
            OUTER=DEFL(1)
         END IF
         IF (NLC==2) THEN
            INNER=DEFL(1)+B2S
            DF2=0.
         END IF
      END IF
   END IF
END IF
EXIT
ELSE
IF (LR<0.) THEN
   NWB=INT((TBHD_LOC(I)-TBHD_LOC(I-1))/WEB_SPC)
   NC(1)=INT((LOC-TBHD_LOC(I-1))/WEB_SPC)
   NC(2)=INT((TBHD_LOC(1)-LOC)/WEB_SPC)
   LW0=MOD(LOC0(CR)-TBHD_LOC(1), WEB_SPC)
   LW0=MIN(LW0, WEB_SPC-LW0)
   IF (LW0>.1) THEN
      IF (INT(-LR0/WEB_SPC)<>NC(2)) THEN
         LBHD_RUP(NLC)=2
         IF (NLC==1) THEN
            OUTER=DEFL(1)
         END IF
         IF (NLC==2) THEN
            INNER=DEFL(1)+B2S
            DF2=0.
         END IF
      END IF
   END IF
END IF
EXIT
END IF
END IF

133
To determine the impact location relative to webs
LWS=WEB_SPC
LW(1)=MOD(LOC-TBHD_LOC(1), WEB_SPC)
LW(2)=LWS-LW(1)
IF (LW(1)<.1.OR.LW(2)<.1) THEN
  IF (LW0>.1) THEN
    LBHD_RUP(NLC)=2
    IF (NLC==1) THEN
      OUTER=DEFL(1)
    END IF
    IF (NLC==2) THEN
      INNER=DEFL(1)+B2S
      DF2=0.
    END IF
  END IF
  IF (LW(1)<.01) LW(1)=.01
  IF (LW(2)<.01) LW(2)=.01
ELSE
  IF (LW(1)<.1) THEN
    IF (NC(1)+NC(2)==NWB) NC(2)=MAX(0, NC(2)-1)
    LW(1)=WEB_SPC+LW(1)
    NC(1)=MAX(0, NC(1)-1)
  ELSE
    IF (NC(1)+NC(2)==NWB) NC(1)=MAX(0, NC(1)-1)
    LW(2)=WEB_SPC+LW(2)
    NC(2)=MAX(0, NC(2)-1)
  END IF
END IF
LWS=2.*WEB_SPC
IF (DEFL(CR)>DW0(CR)) DW0(CR)=DEFL(CR)
END IF
IF (ABS(D_LOC)>FRO*ABS(D_PEN)) THEN
  ! To calculate propagation force and energy
  F1=LBHD_THK(K)*BR(2)*SIGMA(LBHD_MAT(K),1)*SIGN(RFP,D_LOC)
  IF (SHIP_TYPE==2.AND.(NLC==2.OR.NLC==LBHD_NUM) .AND.DEFL(CR)<DF2) F1=0.
  IF (D_LOC<0.) THEN
    N1=1
  ELSE
    N1=2
  END IF
  T(N1)=TN*.5
ELSE
  F1=0.
END IF

! To determine crushed web number
PWF=T*DEFL(CR)/LW
RM=LD_FCT(NLC, PWF, STAT)
PWF=PWF/RM
PWFM=MAX(PWF(1), PWF(2))
IF (LW0<.1.AND.LWS>WEB_SPC.AND..NOT.(DH.AND.CR==2)) THEN
  PSHT=MIN(DEFL(CR), WEB_DEP(K))
  IF (PSHT>DEFL(CR)) THEN
    FW=FW+PWFM*BR(2)
    MOMNT1=MOMNT1-PWFM*SIGN(CFR, D_LOC)*(PSHT+B2S)*BR(2)
    MOMNT2=MOMNT2+PWFM*LOC*BR(2)
    WEB_EN(CR)=WEB_EN(CR)+PWFM*BR(2)*PSHT
  END IF
END IF
THETA=PWF*T*(180./PI)
NC(1)=MIN(NC(1), INT(RM(1)))
NC(2)=MIN(NC(2), INT(RM(2)))
DWF=0.
DO I=1, 2
  DO WHILE (NC(I)>0)
    DWFT=WEB_SPC/(LW(I)+NC(I)*WEB_SPC)*
    & (DEFL(CR)-PWF(I)/T(I)*(NC(I)*LW(I)
    & +.5*WEB_SPC*(NC(I)-1)*NC(I))
    IF (DWFT<0.) THEN
      IF (NC(I)==1.AND.I==(3-N10).AND.LW(O)<.1
        .AND.LW(1)>=.1.AND.LW(2)>=.1) THEN
        DWF(I,1)=DW0(CR)
        EXIT
      ELSE
        NC(I)=NC(I)-1
        END IF
    ELSE
      DWF(I,NC(I))=DWFT
      EXIT
    END IF
  END DO
END DO
!
To calculate the maximum deflection and elongation before rupture
WA=DEFL_LIM(N1, NC, LW, WEB_SPC, SIGMA(LBHD_MAT(K),3), THETA)
DEFLA=MAX(MIN(WA, DEFL(CR)), DEFL(CR)-D_PEN)
!
If the deflection exceed the limits, re-set flags
IF (DEFL(CR)>WA.AND.(NC(1)>0.OR.NC(2)>0)) THEN
  J=0
  DO I=1, 2
    DO WHILE (NC(I)>0)
      DWFT=WEB_SPC/(LW(I)+NC(I)*WEB_SPC)*
      & (DEFLA-PWF(I)/T(I)*(NC(I)*LW(I)
      & +.5*WEB_SPC*(NC(I)-1)*NC(I))
      IF (DWFT<0.) THEN
        IF (NC(I)==1.AND.I==(3-N10).AND.LW(O)<.1
          .AND.LW(1)>=.1.AND.LW(2)>=.1) THEN
          DWF(I,1)=DW0(CR)
          EXIT
        ELSE
          NC(I)=NC(I)-1
          J=1
        END IF
      ELSE
        DWF(I,NC(I))=DWFT
        EXIT
      END IF
    END DO
  END DO
  IF (J==1) THEN
    WA=DEFL_LIM(N1, NC, LW, WEB_SPC, SIGMA(LBHD_MAT(K),3), THETA)
    DEFLA=MAX(MIN(WA, DEFL(CR)), DEFL(CR)-D_PEN)
  END IF
END IF
END IF
IF (DEFL(CR)>WA) THEN
  LBHD_RUP(NLC)=2
  IF (NLC==1) THEN
    OUTER=DEFLA
  END IF
  IF (NLC==2) THEN
    INNER=DEFLA+B2S
    DF2=0.
END IF
END IF
!
To calculate membrane tension energy absorbed in the time step
! calculate deflection at webs and elongations
ET=0.
ET2=0.
DO I=1, 2
PCS=PWF(I)*WEB_SPC/T(I)
DO J=NC(I), 1, -1
NN=NC(I)-J
PSHT=PSHWEB(I, J, NLC, LW, WEB_SPC)
DWF(I,J)=(NN+1)*DWF(I,NC(I))+.5*PCS*NN*(NN+1)
IF (J==1.AND.I==(3-N10).AND.LWO<.1.AND.LWS==WEB_SPC
& .AND..NOT.(DH.AND.CR==2)) THEN
IF (PSHT>=MAX(DWF(I,1), DW0(CR))) THEN
DWO(CR)=PSHT
MOMNT1=MOMNT1-PWFM*SIGN(CFR,D_LOC)*(PSHT+B2S)
& *BR(2)
MOMNT2=MOMNT2+PWFM*BR(2)*(LOC+SIGN(LW(I)+(J-1)
& *WEB_SPC, I-1.5))
END IF
DWF(I,1)=MAX(DWF(I,1), DW0(CR))
WEB_EN(CR)=WEB_EN(CR)+BR(2)*(PWFM*DWO(CR)
& +PWF(I)*(DWF(I,1)-DWO(CR)))
ELSE
IF (PSHT>=DWF(I,J)) THEN
FW=FW+PWFM*BR(2)
MOMNT1=MOMNT1-PWFM*SIGN(CFR,D_LOC)*(PSHT+B2S)
& *BR(2)
MOMNT2=MOMNT2+PWFM*BR(2)*(LOC+SIGN(LW(I)+(J-1)
& *WEB_SPC, I-1.5))
DWF(I,J)=PSHT
WEB_EN(CR)=WEB_EN(CR)+BR(2)*PWFM*PSHT
ELSE
WEB_EN(CR)=WEB_EN(CR)+BR(2)*PWF(I)*DWF(I,J)
END IF
END IF
IF (J==NC(I)) THEN
ET(I)=ET(I)+EL(DWF(I,J), 0., WEB_SPC)
IF (DH.AND.STAT==0)
& ET2=ET2+EL(DWF(I,J), 0., WEB_SPC)
ELSE
ET(I)=ET(I)+EL(DWF(I,J), DWF(I,J+1), WEB_SPC)
IF (DH.AND.STAT==0)
& ET2=ET2+EL(DWF(I,J), DWF(I,J+1), WEB_SPC)
END IF
END DO
ET(I)=ET(I)+EL(DEFLA, DWF(I,1), LW(I))
IF (F1>0..AND.I<n1) F2=F2+F1*(DEFLA-DWF(I,1))/LW(I)
END DO
IF (DH.AND.STAT==0) THEN
IF (LWS>WEB_SPC) THEN
BA2=.5*(ATAND((DEFLA-DWF(1,1))/WEB_SPC)
& +ATAND((DEFLA-DWF(2,1))/WEB_SPC))
DF2=DEFLA
IF (BA2<SIGMA(LBHD_MAT(K2), 3)) THEN
ET2=ET2+EL(DEFLA, DWF(1,1), WEB_SPC)
& +EL(DEFLA, DWF(2,1), WEB_SPC)
ELSE
LBHD_RUP(NLC+1)=2
IF (NLC==1) THEN
INNER=DEFLA
DF2=0.
END IF
END IF
ELSE
BA2=.5*(ATAND(ABS(DWF(2,1)-DWF(1,1))/WEB_SPC)
&    +MAX(ATAND((DWF(1,1)-DWF(1,2))/WEB_SPC),
&    +ATAND((DWF(2,1)-DWF(2,2))/WEB_SPC))
IF (REACHED==2) THEN
DEFLA2=MAX(DEFL(2)-(DEFL(1)-DEFLA),
&    DWF(1,1)+(DWF(2,1)-DWF(1,1))*LW(1)/LWS)
DF2=DEFLA2
IF (DEFLA2==DEFL(2)-(DEFL(1)-DEFLA)) BA2=0.
IF (BA2<SIGMA(LBHD_MAT(K2),3)) THEN
    ET2=ET2+EL(DEFLA2, DWF(1,1), LW(1))
    +EL(DEFLA2, DWF(2,1), LW(2))
ELSE
LBHD_RUP(NLC+1)=2
IF (NLC==1) THEN
    INNER=DEFLA
    DF2=0.
END IF
END IF
ELSE
DF2=DWF(1,1)+(DWF(2,1)-DWF(1,1))*LW(1)/LWS
IF (BA2<SIGMA(LBHD_MAT(K2),3)) THEN
    ET2=ET2+EL(DWF(1,1), DWF(2,1), LWS)
ELSE
LBHD_RUP(NLC+1)=2
IF (NLC==1) THEN
    INNER=DEFLA
    DF2=0.
END IF
END IF
END IF
! calculate absorbed energy
MEMB_EN(CR)=WEB_EN(CR)+BR(4)*(T(1)*ET(1)+T(2)*ET(2))
FORC(1)=FORC(1)+F1
IF (DH.AND.STAT==0) THEN
    IF (BA2<SIGMA(LBHD_MAT(K2),3)) THEN
        MEMB_EN(2)=MEMB_EN(2)+TN2*BR(4)*ET2
    ELSE
        MEMB_EN(2)=MEMB_EN0(2)
    END IF
    REACHED=1
    EXIT
END IF
END DO
! Average membrane tension force
IF (ABS(FL_MAX)<ABS(FORC(1))) FORC(1)=FL_MAX
MISC_EN=FORC(1)*D_LOC
LOST_EN=MAX(0., MEMB_EN(1)+MEMB_EN(2)-MEMB_EN0(1)-MEMB_EN0(2)
&    -MISC_EN)
IF (ABS(D_LOC)>ABS(D_PEN)*FRO.AND.FORC(1)<FL_MAX) THEN
    FORC(2)=MAX(F2, LOST_EN/(ABS(CFR*D_LOC)+D_PEN)-FW)
END IF
! to include friction force term
IF (FORC(2)>0.) THEN
    FORC(1)=FORC(1)+FORC(2)*SIGN(CFR, D_LOC)
ELSE
    FORC(2) = 0.
END IF
MOMNT1 = MOMNT1 + FORC(1) * (DEFL(1) + BEAM1 *.5 - LBHD_LOC(LBHD_CUR))
MOMNT2 = MOMNT2 + FORC(2) * LOC
FORC(1) = FORC(1) + (FW + FWC) * CFR
FORC(2) = FORC(2) + FW + FWC
IF (ABS(FL_MAX) < ABS(FORC(1))) THEN
    MOMNT1 = MOMNT1 * FL_MAX / FORC(1)
    FORC(1) = FL_MAX
END IF
MOMNT = MOMNT1 + MOMNT2
MISC_EN = FORC(1) * D_LOC
ELSE
    IF (D_PEN <> 0.) THEN
        FORC(2) = MAX(F2, LOST_EN / ABS(D_PEN) - FW - FWC)
        MOMNT = MOMNT2 + FORC(2) * LOC
        FORC(2) = FORC(2) + FW + FWC
        FORC(1) = 0.
    ELSE
        FORC = 0.
    END IF
MISC_EN = 0.
END IF
ANGLE = ATAN2D(FORC(2), FORC(1))
FORCE = V_ABS(FORC)
!
! test print
IF (PRESENT(M)) THEN
    IF (M < 3) WRITE (23, '(4X, 2F20.1)') LOST_EN + MISC_EN, &
        MAX(MEMB_EN0(1) + MEMB_EN0(2), MEMB_EN(1) + MEMB_EN(2)) + MISC_EN
END IF
!
! If the deflection exceed the limits, re-set flags and current bulkhead
IF (REACHED > 0. AND. LBHD_RUP(LBHD_CUR) > 1) THEN
    DEFL(1) = DEFL(2)
    IF (DH) THEN
        MEMB_EN(1) = MEMB_EN(2) + WEB_EN(1)
        MEMB_EN0(1) = MEMB_EN0(2) + WEB_EN(1)
    ELSE
        MEMB_EN(1) = MEMB_EN(2)
        MEMB_EN0(1) = MEMB_EN0(2)
    END IF
    LOC0(1) = LOC0(2)
    DW0(1) = DW0(2)
    DEFL(2) = 0.
    MEMB_EN(2) = 0.
    MEMB_EN0(2) = 0.
    LOC0(2) = 0.
    REACHED = REACHED - 1
    LBHD_CUR = LBHD_CUR + 1
IF (REACHED > 0. AND. LBHD_RUP(LBHD_CUR) > 1) THEN
    DEFL = 0.
    MEMB_EN = 0.
    MEMB_EN0 = 0.
    LOC0 = 0.
    DW0 = 0.
    REACHED = 0
    LBHD_CUR = LBHD_CUR + 1
END IF
ELSE
    IF (REACHED > 0. AND. LBHD_RUP(LBHD_CUR + 1) > 1) THEN
        DEFL(2) = 0.
MEMB_EN(2)=0.
MEMB_EN0(2)=0.
LOC0(2)=0.
DW0(2)=0.
REACHED=1
END IF
END IF
DO I=1, 2
MEMB_EN0(I)=MAX(MEMB_EN0(I), MEMB_EN(I))
END DO
!
END SUBROUTINE MEMB_FORCE
!
!* Function PEN_BOW is to determine the point coordinates of penetrating bow
! Input : LOC - the location of the impact point along the ship length
! (measured from FP)
! PEN - the depth of penetration across the ship (transversely)
! OMEGA - collision angle at this step (OMEGA1-OMEGA2)
! HEA - half-entrance angle of striking ship
! BEAM - breadth of the struck ship
! Output: BP - coordinates of penetrating bow
!
FUNCTION PEN_BOW(LOC, PEN, OMEGA, HEA, BEAM) RESULT(BP)
USE GENERAL, ONLY : V_ABS
!
IMPLICIT NONE
REAL, INTENT (IN) :: LOC, PEN, OMEGA, HEA, BEAM
REAL, DIMENSION (5,2) :: BP
!
SBA - side length of penetrating wedge
SBL - maximum side length of penetrating wedge for ship beam width
REAL :: SBA, SBL
!
If penetrating bow is wedge shape
BP=0.
BP(3,1)=LOC
BP(3,2)=PEN
IF (OMEGA==HEA) THEN
  BP(2,1)=LOC
ELSE
  BP(2,1)=LOC+PEN/TAND(OMEGA+HEA)
END IF
IF (OMEGA==HEA) THEN
  BP(4,1)=LOC
ELSE
  BP(4,1)=LOC+PEN/TAND(OMEGA-HEA)
END IF
BP(1,1)=BP(2,1)
BP(5,1)=BP(4,1)
SBL=0.5*BEAM/SIND(HEA)
!
If the parallel body is involved
SBA=V_ABS(BP(2,:)-BP(3,:))
IF (SBA>SBL) THEN
  BP(2,1)=LOC+SBL*COSD(OMEGA+HEA)
  BP(2,2)=PEN-SBL*SIND(OMEGA+HEA)
  BP(1,1)=BP(2,1)-BP(2,2)/TAND(OMEGA)
END IF
!
SBA=V_ABS(BP(4,:)-BP(3,:))
IF (SBA>SBL) THEN
  BP(4,1)=LOC+SBL*COSD(OMEGA-HEA)
  BP(4,2)=PEN-SBL*SIND(OMEGA-HEA)

BP(5,1)=BP(4,1)-BP(4,2)/TAND(OMEGA)
END IF
!
END FUNCTION PEN_BOW
!
!* Function REL_MOVE is to calculate relative movement of striking bow
! Output: MV - relative movement of striking bow
!
FUNCTION REL_MOVE() RESULT(MV)
USE GENERAL, ONLY : V_ABS
IMPLICIT NONE
!
REAL, DIMENSION (2) :: MV
!
INTEGER :: I, N
REAL, DIMENSION (5,2) :: BPS
REAL, DIMENSION (2) :: V1, V2
REAL :: R1, R2
!
BPS=BP2
DO I=1, 2
  V1=BP1(3-I,:)-BP1(3-I+1,:)
  V2=BP2(3-I,:)-BP2(3-I+1,:)
  R1=V_ABS(V1)
  R2=V_ABS(V2)
  IF (R2==0.) THEN
    BPS(3-I,:)=BPS(3-I+1,:)+V1
  ELSE
    BPS(3-I,:)=BPS(3-I+1,:)+V2*R1/R2
  END IF
  V1=BP1(3+I,:)-BP1(3+I-1,:)
  V2=BP2(3+I,:)-BP2(3+I-1,:)
  R1=V_ABS(V1)
  R2=V_ABS(V2)
  IF (R2==0.) THEN
    BPS(3+I,:)=BPS(3+I-1,:)+V1
  ELSE
    BPS(3+I,:)=BPS(3+I-1,:)+V2*R1/R2
  END IF
END DO
!
N=0
MV=0.
DO I=1, 4
  V1=BPS(I,:)-BP1(I,:)
  IF (V1(2)==0..AND.BPS(I,2)==0.) CONTINUE
  N=N+1
  MV=MV+V1
END DO
MV=MV/N
!
END FUNCTION REL_MOVE
!
!* Subroutine MINO_DECK is to calculate the reaction force and moment
! caused by Minorsky mechanism on decks in LOC-PEN coordinate system
! Input : THK - deck plate thickness
!   WDX - extent of deck opening
! Output: FORCE - reaction force in LOC-PEN coordinate system
!   MOMNT - reaction moment about the origin of LOC-PEN system
!   MINO_EN - absorbed energy during the time step (for debug only)
!
SUBROUTINE MINO_DECK(THK, FORCE, MOMNT, MINO_EN, WDX)
USE GENERAL
! IMPLICIT NONE
REAL, INTENT (IN) :: THK
REAL, DIMENSION (2), OPTIONAL, INTENT (IN) :: WDX
REAL, DIMENSION (2), INTENT (OUT) :: FORCE
REAL, INTENT (OUT) :: MOMNT, MINO_EN
!
! VEC0 - vector of profile segment of penetrating bow before the step
! VECL - vector of profile segment of penetrating bow after the step
! VEC1 - movement of the first node during the step
! VEC2 - movement of the second node during the step
! ANG* - argument of VEC* in LOC-PEN system
! A_V* - absolute values of VEC*
REAL, DIMENSION (2) :: VEC0, VECL, VEC1, VEC2
REAL :: ANG0, ANG1, ANG2, A_V0, A_VL
!
! NPT - nodes of protruding area during the step
! AR_S - deck area damaged by one profile segment
! CTR - force acting point in LOC-PEN system
! MOVE - average sweeping of one segment in LOC-PEN system
! A_MV - absolute value of sweeping distance of one segment
! D_MV - sweeping direction of one segment in LOC-PEN system
! A_FS - reaction force (area/move) on one segment
! FC_S - force components (area/move) on one segment in LOC-PEN system
REAL, DIMENSION (4, 2) :: NPT
REAL :: AR_S, A_MV, D_MV, A_FS
REAL, DIMENSION (2) :: CTR, MOVE, FC_S
REAL, DIMENSION (2) :: PI0, PI1, PL0, PL1
!
INTEGER :: I, ND
LOGICAL :: STRGR
!
FORCE=0.
MOMNT=0.
STRGR=.FALSE.
!
To determine whether striking bow has penetrated whole stringer
IF (PRESENT(WDX).AND.(PI0(2)>=WDX(1).OR.PI1(2)>=WDX(1)
& .OR.PL0(2)>=WDX(1).OR.PL1(2)>=WDX(1))) STRGR=.TRUE.
!
To calculate absorbed energy for debugging purpose
MINO_EN=0.
!
To determine deck area damaged in each profile segment of penetrating bow
DO I=1, 4
!
To calculate vectors of profile segment and its movement
PI0=BP1(I,::)
PI1=BP1(I+1,::)
PL0=BP2(I,::)
PL1=BP2(I+1,::)
VEC0=PI1-PI0
VECL=PL1-PL0
VEC1=PL0-PI0
VEC2=PL1-PI1
A_V0=V_ABS(VEC0)
A_VL=V_ABS(VECL)
MOVE=0.
!
To calculate outwards swept area and average movement
if initial segment length is zero
IF (A_V0==0.) THEN
if length of the segment or node movement is keeping zero
IF (V_ABS(VEC1)*V_ABS(VEC2)*A_VL==0.) THEN
  AR_S=0.
  CTR=0.
ELSE
  ANG1=ATAN2D(VEC1(2), VEC1(1))
  ANG2=ATAN2D(VEC2(2), VEC2(1))
  IF (I<3.AND.(ANG1>90.).AND.(ANG2<ANG1))
    & .OR.(I>=3.AND.(ANG2<90.).AND.(ANG2<ANG1))) THEN
    NPT(1,:)=PI0
    NPT(2,:)=PL0
    NPT(3,:)=PL1
    IF (STRGR) THEN
      CALL AR_CTR(3, NPT, AR_S, CTR, WDX)
    ELSE
      CALL AR_CTR(3, NPT, AR_S, CTR)
    END IF
    IF (AR_S<0.) THEN
      AR_S=0.
      CTR=0.
    ELSE
      IF (I>=3) THEN
        MOVE=VEC1*0.5
      ELSE
        MOVE=VEC2*0.5
      END IF
    END IF
  ELSE
    AR_S=0.
    CTR=0.
  END IF
ENDIF
ELSE
  the initial length is not zero
ELSE
  calculate the moving direction of each node
  ANG0=ATAN2D(VEC0(2), VEC0(1))
  IF (V_ABS(VEC1)==0.) THEN
    ANG1=ANG0
  ELSE
    ANG1=ATAN2D(VEC1(2), VEC1(1))
    IF (I<3.AND.ANG1<0.) ANG1=360.+ANG1
  END IF
  IF (V_ABS(VEC2)==0.) THEN
    ANG2=ANG0
  ELSE
    ANG2=ATAN2D(VEC2(2), VEC2(1))
    IF (I<3.AND.ANG2<0.) ANG2=360.+ANG2
  END IF
  IF ((ANG1-ANG0<180.).AND.(ANG1-ANG0>0.)) THEN
    NPT(1,:)=PI0
    NPT(2,:)=PL0
  ELSE
    NPT(3,:)=PL1
  END IF
ENDIF
NPT(4,:)=PI1
IF (STRGR) THEN
    CALL AR_CTR(4, NPT, AR_S, CTR, WDX)
ELSE
    CALL AR_CTR(4, NPT, AR_S, CTR)
END IF
IF (I>=3) THEN
    IF (A_VL==0.) THEN
        MOVE=(VEC1+PL0+VEC0-PI1)*0.5
    ELSE
        MOVE=(VEC1+PL0+VECL*(A_V0/A_VL)-PI1)*0.5
    END IF
ELSE
    IF (A_VL==0.) THEN
        MOVE=(VEC2+PL1-VEC0-PI0)*0.5
    ELSE
        MOVE=(VEC2+PL1-VECL*(A_V0/A_VL)-PI0)*0.5
    END IF
END IF
!
!    if the second node is moving inwards
ELSE
    NPT(3,:)=INTCPT(PI0, PI1, PL0, PL1)
    IF (STRGR) THEN
        CALL AR_CTR(3, NPT, AR_S, CTR, WDX)
    ELSE
        CALL AR_CTR(3, NPT, AR_S, CTR)
    END IF
    IF (I>=3) THEN
        IF (A_VL==0.) THEN
            MOVE=(PL0+VEC0-PI1)*0.5
        ELSE
            MOVE=(PL0+VECL*(A_V0/A_VL)-PI1)*0.5
        END IF
    ELSE
        MOVE=VEC2*0.5
    END IF
!    if the second node is moving outwards too
IF ((ANG2-ANG0<180.).AND.(ANG2-ANG0>0.)) THEN
    NPT(1,:)=INTCPT(PI0, PI1, PL0, PL1)
    NPT(2,:)=PL1
    NPT(3,:)=PI1
    IF (STRGR) THEN
        CALL AR_CTR(3, NPT, AR_S, CTR, WDX)
    ELSE
        CALL AR_CTR(3, NPT, AR_S, CTR)
    END IF
    IF (I>=3) THEN
        IF (A_VL==0.) THEN
            MOVE=(PL0+VEC0-PI1)*0.5
        ELSE
            MOVE=(PL0+VECL*(A_V0/A_VL)-PI1)*0.5
        END IF
    ELSE
        MOVE=VEC2*0.5
    END IF
!    if the second node is moving inwards
ELSE
    !    if the first node is moving inwards
    END IF
END IF
ELSE
    AR_S=0.
    CTR=0.
END IF
END IF
END IF

! To accumulate the area components and area moment
A_MV=V_ABS(MOVE)
IF (A_MV*AR_S<1.E-10) THEN
    FC_S=0.
ELSE
    D_MV=ATAN2D(MOVE(2), MOVE(1))
    A_FS=AR_S/A_MV
    FC_S(1)=A_FS*COSD(D_MV)
    FC_S(2)=A_FS*SIND(D_MV)
END IF
FORCE=FORCE+FC_S
MOMNT=MOMNT-FC_S(1)*CTR(2)+FC_S(2)*CTR(1)

! calculate absorbed energy for debugging purpose
MINO_EN=MINO_EN+AR_S

END DO

! To finalize the results for deck
FORCE=FORCE*DKE*THK
MOMNT=MOMNT*DKE*THK
MINO_EN=MINO_EN*DKE*THK

END SUBROUTINE MINO_DECK

!* Subroutine MINO_SIDE is to calculate the reaction force and moment
! caused by Minorsky mechanism in side shell and longitudinal bulkheads
! in LOC-PEN coordinate system
! Input : SHIP_STR module
! Output: FORCE - reaction force caused by Minorsky mechanism
! MOMNT - reaction moment about the origin of LOC-PEN system
! MINO_EN - absorbed energy during the time step (for debug only)
!
SUBROUTINE MINO_SIDE(FORCE, MOMNT, MINO_EN)
USE SHIP_DIM
USE SHIP_STR

IMPLICIT NONE
REAL, INTENT (OUT) :: FORCE, MOMNT, MINO_EN

INTEGER :: I, K
REAL :: DEPTH, ARM, FC, PBD, M0, MB, P, CD

! To calculate Minorsky's mechanism on shell and Longitudinal bulkheads
FORCE=0.
MOMNT=0.
MINO_EN=0.
DEPTH=EFF_DP()
PBD=(BP1(3,2)/SIND(OMEGAR(1))+BP2(3,2)/SIND(OMEGAR(2)))*.5
M0=BP2(3,1)-BP1(3,1)+PBD*(COSD(OMEGAR(2))-COSD(OMEGAR(1)))
MB=BP2(3,1)-BP1(3,1)
P=(BP1(3,2)+BP2(3,2))*.5
DO I=1, lBHD_NUM
    ARM=BEAM1*.5-LBHD_LOC(I)
    IF (ARM>BP2(3,2)) EXIT
    IF (LBHD_RUP(I)<2) CYCLE

    ! Calculate Minorsky mechanism
    AR_S=DEPTH*ARM
    CTR=0.
    END IF
    END IF
    END IF

    ! To accumulate the area components and area moment
    A_MV=V_ABS(MOVE)
    IF (A_MV*AR_S<1.E-10) THEN
        FC_S=0.
    ELSE
        D_MV=ATAN2D(MOVE(2), MOVE(1))
        A_FS=AR_S/A_MV
        FC_S(1)=A_FS*COSD(D_MV)
        FC_S(2)=A_FS*SIND(D_MV)
    END IF
    FORCE=FORCE+FC_S
    MOMNT=MOMNT-FC_S(1)*CTR(2)+FC_S(2)*CTR(1)

    ! calculate absorbed energy for debugging purpose
    MINO_EN=MINO_EN+AR_S

    END DO

    ! To finalize the results for deck
    FORCE=FORCE*DKE*THK
    MOMNT=MOMNT*DKE*THK
    MINO_EN=MINO_EN*DKE*THK

    END SUBROUTINE MINO_DECK

!* Subroutine MINO_SIDE is to calculate the reaction force and moment
! caused by Minorsky mechanism in side shell and longitudinal bulkheads
! in LOC-PEN coordinate system
! Input : SHIP_STR module
! Output: FORCE - reaction force caused by Minorsky mechanism
! MOMNT - reaction moment about the origin of LOC-PEN system
! MINO_EN - absorbed energy during the time step (for debug only)
!
SUBROUTINE MINO_SIDE(FORCE, MOMNT, MINO_EN)
USE SHIP_DIM
USE SHIP_STR

IMPLICIT NONE
REAL, INTENT (OUT) :: FORCE, MOMNT, MINO_EN

INTEGER :: I, K
REAL :: DEPTH, ARM, FC, PBD, M0, MB, P, CD

! To calculate Minorsky's mechanism on shell and Longitudinal bulkheads
FORCE=0.
MOMNT=0.
MINO_EN=0.
DEPTH=EFF_DP()
PBD=(BP1(3,2)/SIND(OMEGAR(1))+BP2(3,2)/SIND(OMEGAR(2)))*.5
M0=BP2(3,1)-BP1(3,1)+PBD*(COSD(OMEGAR(2))-COSD(OMEGAR(1)))
MB=BP2(3,1)-BP1(3,1)
P=(BP1(3,2)+BP2(3,2))*.5
DO I=1, lBHD_NUM
    ARM=BEAM1*.5-LBHD_LOC(I)
    IF (ARM>BP2(3,2)) EXIT
    IF (LBHD_RUP(I)<2) CYCLE
IF (ARM>P) THEN
  CD=MB*.5
ELSE
  CD=M0+(MB-M0)*ARM/P
END IF

K=LBHD_NUM/2
IF (K<LBHD_NUM*.5) K=K+1
IF (I<=K) THEN
  K=I
ELSE
  K=K*2-I
END IF

FC=DEPTH*LBHD_THK(K)*SIGN(DKE, CD)
FORCE=FORCE+FC
MOMNT=MOMNT-FC*ARM
MINO_EN=MINO_EN+FC*CD
END DO

END SUBROUTINE MINO_SIDE

!* Subroutine MINO_FORCE is to calculate the reaction force and moment
!* caused by Minorsky mechanism in LOC-PEN coordinate system
!* Input : D_LOC - shift of impact point along the length in each time step
!*   D_PEN - increment of penetration during each time step
!*   M - total number of scenarios (for testing output)
!* Output: FORCE - reaction force caused by Minorsky mechanism
!*   ANGLE - striking force angle in LOC-PEN system
!*   MOMNT - reaction moment about the origin of LOC-PEN system
!* SUBROUTINE MINO_FORCE(D_LOC, D_PEN, FORCE, ANGLE, MOMNT, M)
USE GENERAL
USE SHIP_DIM
USE SHIP_STR

IMPLICIT NONE
INTEGER, OPTIONAL, INTENT (IN) :: M
REAL, INTENT (IN) :: D_LOC, D_PEN
REAL, INTENT (OUT) :: FORCE, ANGLE, MOMNT

REAL, DIMENSION (2) :: FORCE_D, FORCE_S, WDX, FORC
REAL :: MOMNT_D, MINO_D, MOMNT_S, MINO_S
REAL :: FORCE_L, MOMNT_L, MINO_L, MINO_EN

!* To calculate Minosky's mechanism in decks
CALL MINO_DECK(TDAB_THK, FORCE_D, MOMNT_D, MINO_D)
IF (TSTR_THK>0.) THEN
  WDX(1)=STRG_WID
  WDX(2)=BEAM1-STRG_WID
  CALL MINO_DECK(TSTR_THK, FORCE_S, MOMNT_S, MINO_S, WDX)
ELSE
  FORCE_S=0.
MOMNT_S=0.
MINO_S=0.
END IF
FORC=FORCE_D+FORCE_S
MOMNT=MOMNT_D+MOMNT_S
!
To calculate Minosky's mechanism in side shells and longitudinal bulkheads
IF (ABS(D_LOC/D_PEN)>FRO) THEN
   CALL MINO_SIDE(FORCE_L, MOMNT_L, MINO_L)
   FORC(1)=FORC(1)+FORCE_L
   MOMNT=MOMNT+MOMNT_L
END IF
FORCE=V_ABS(FORC)
ANGLE=ATAN2D(FORC(2),FORC(1))
!
calculate absorbed energy for debugging purpose
IF (PRESENT(M)) THEN
   IF (M<3) THEN
      MINO_EN=MINO_D+MINO_S+MINO_L
      MINO_EN0=MINO_EN0+MINO_EN
      WRITE (23,'(4X,2F20.1)') MINO_EN, MINO_EN0
   END IF
END IF
!
END SUBROUTINE MINO_FORCE
!
END MODULE COL_FORCE

A.3.4 initiate.for

!!! INITIATE.FOR     by Donghui Chen
!!!
!!! VERSION: 2.1     DATE: December 10, 1999
!!!
!!! REVISION NOTES:
!!! v2.1
!!! - Incorporate definite bow depth
!!! v2.0
!!! - Incorporate Double Hull Tankers
!!! - Rename INI-SINGLE to INI_SHIP and revise the subroutine accordingly
!!! v1.2
!!! - Introduce properties of web frames into module SHIP_STR
!!! v1.1
!!! - Separate out from COLLISION.FOR
!!! - Initiate data from input files
!!! STRUCTURE.IN - data of structure and arrangement
!!! COLLISION.IN - data of collision scenario
!!! - Introduce material grade, 1 - MS, 2 - HT32, 3 - HT36
!!!
!!! CONTENTS:
!!! - SUBROUTINES
!!!   INI_SHIP - initiates variables for tankers
!!!   INI_COLLIS - initiates variables for collision scenario
!!!
!!! Subroutine INI_SHIP is to initiate the variables for tankers
!!! Output: common blocks /STRUCK_DIM/, /STRIKE_DIM, /CONSTRUCTION/,
SUBROUTINE INI_SHIP
!
USE SHIP_DIM
USE SHIP_STR
!
IMPLICIT NONE
!
K - number of side shell/bulkheads at one side (include centerline bulkhead)
INTEGER :: I, J, K, L
REAL :: DDT, DDB
!
In future, the half entrance angle of the striking bow and displacements or drafts will be初始ated by a variable generator of Monte Carlo method
!
Open input file
OPEN (UNIT=01, FILE="structure.in")
!
To initiate data of data block STRUCK_DIM
!
READ (01,*) SHIP_TYPE
READ (01,*) LBP1, BEAM1, DEPTH1, DRAFT1, DISP1, WEB_SPC
A1=ANN(BEAM1, DRAFT1, DISP1, LBP1)
JV1=J_VIRT(LBP1, DRAFT1, DISP1)
!
To initiate data of data block STRIKE_DIM
!
READ (01,*) LBP2, BEAM2, DRAFT2, DISP2, BOW_HT, HEA
A2=ANN(BEAM2, DRAFT2, DISP2, LBP2)
JV2=J_VIRT(LBP2, DRAFT2, DISP2)
!
To initiate data of data block CONSTRUCTION
!
READ (01,*) TBHD_NUM
IF (TBHD_NUM>12) THEN
   PRINT*, "ERROR: TBHD_NUM to be less than 12."
END IF
TBHD_LOC=0.
READ (01,*) (TBHD_LOC(I), I=1, TBHD_NUM)
TBHD_LOC=TBHD_LOC-.5*LBP1
!
READ (01,*) LBHD_NUM
IF (LBHD_NUM>7) THEN
   PRINT*, "ERROR: LBHD_NUM to be less than 7."
END IF
K=LBHD_NUM/2
IF (K<LBHD_NUM*.5) K=K+1
LBHD_THK=0.
LBHD_MAT=0.
READ (01,*) (LBHD_LOC(I), LBHD_THK(I), LBHD_MAT(I), I=1, K)
I=LBHD_NUM
DO WHILE (I>K)
   LBHD_LOC(I)=-LBHD_LOC(LBHD_NUM-I+1)
   I=I-1
END DO
!
DDT=BOW_HT-DRAFT2+DRAFT1
DDB=DRAFT1-DRAFT2
TDAB_THK=0.
IF (SHIP_TYPE==1) THEN
   READ (01,*) DECK_THK, BTM_THK
   READ (01,*) STRG_NUM, STRG_WID
ELSE
    READ (01,*) DECK_THK, IBTM_THK, BTM_THK, DBL_HT
    IF (DBL_HT<=DDT.AND.DBL_HT>=DDB) TDAB_THK=TDAB_THK+IBTM_THK
    READ (01,*) STRG_NUM
    STRG_WID=LBHD_LOC(1)-LBHD_LOC(2)
END IF
IF (DEPTH1<=DDT) TDAB_THK=TDAB_THK+DECK_THK
IF (DDB<=0.) TDAB_THK=TDAB_THK+BTM_THK
STRG_LOC=0.
STRG_THK=0.
TSTR_THK=0.
IF (STRG_NUM>0) THEN
    READ (01,*) (STRG_LOC(I), I=1, STRG_NUM)
    READ (01,*) (STRG_THK(I), I=1, STRG_NUM)
    DO I=1, STRG_NUM
        IF (STRG_LOC(I)<=DDT.AND.STRG_LOC(I)>=DDB)
            TSTR_THK=TSTR_THK+STRG_THK(I)
        END IF
    END DO
END IF
*
* To initiate data of data block WEBS
*
WEB_SUP=0
WEB_DEP=0.
WEB_MAT=0
WEB_THK=0.
WEB_SMZ=0.
WEB_SPN=0.
WEB_GAP=0.
SUP_MAT=0.
SUP_ARA=0.
SUP_GRA=0.
SUP_LEN=0.
STF_THK=0.
STF_GRA=0.
IF (SHIP_TYPE==2) THEN
    L=2
ELSE
    L=1
END IF
DO I=L, K
    READ (01,*) WEB_SUP(I), WEB_DEP(I), WEB_STF(I)
    IF (WEB_SUP(I)>4) THEN
        PRINT*, "ERROR: WEB_SUP to be less than 4."
        END IF
    READ (01,*) (WEB_MAT(I,J), WEB_THK(I,J), WEB_SMZ(I,J),
        J=1, WEB_SUP(I))
    READ (01,*) (WEB_SPN(I,J), J=1, WEB_SUP(I)-1)
    READ (01,*) (WEB_GAP(I,J), J=1, WEB_SUP(I)-1)
    IF (WEB_SUP(I)>2) THEN
        READ (01,*) (SUP_MAT(I,J), SUP_ARA(I,J), SUP_GRA(I,J),
            SUP_LEN(I,J), J=2, WEB_SUP(I)-1)
    END IF
    READ (01,*) STF_THK(I), STF_GRA(I)
END DO
IF (SHIP_TYPE==2) THEN
    WEB_SUP(1)=WEB_SUP(2)
    WEB_DEP(1)=WEB_DEP(2)
    WEB_STF(1)=WEB_STF(2)
WEB_MAT(1,:) = WEB_MAT(2,:)
WEB_THK(1,:) = WEB_THK(2,:)
WEB_SMZ(1,:) = WEB_SMZ(2,:)
WEB_SPN(1,:) = WEB_SPN(2,:)
SUP_MAT(1,:) = SUP_MAT(2,:)
SUP_ARA(1,:) = SUP_ARA(2,:)
SUP_GRA(1,:) = SUP_GRA(2,:)
SUP_LEN(1,:) = SUP_LEN(2,:)
STF_THK(1) = STF_THK(2)
STF_GRA(1) = STF_GRA(2)
END IF
!
LD_SPN = LOAD()
LD_BND = A_BND()
LD_SHR = A_SHR()
LD_CMP = A_CMP()
LD_CRS = A_CRS()
!
CLOSE (01)
!
END SUBROUTINE INI_SHIP
!
** Subroutine INI_COLLIS is to initiate the variables of collision scenario
! Output: common blocks /STRUCK_COL/ and /STRIKE_COL/
**
SUBROUTINE INI_COLLIS
!
USE GENERAL, ONLY : KTM
USE SHIP_DIM
USE SHIP_COL
!
IMPLICIT NONE
!
! VEL1 - speed of the struck ship (knots)
! VEL2 - speed of the striking ship (knots)
! L - impact point location measured after of amidship of the struck ship
! PHI - collision angle
REAL :: VEL1, VEL2, L, PHI
!
!* To generate the initial collision scenario
!* This part will be replaced by a variable generator of Monte Carlo method
!
READ (02,* ) VEL1, VEL2, L, PHI
!
!* To initiate data of data block STRUCK_COL
!
X1 = (/0., 0./)
OMEGA1 = 0.
W1 = 0.
PEN = 0.
V1(1) = VEL1*KTM
V1(2) = 0.
LOC = L
!
!* To initiate data of data block STRIKE_COL
!
OMEGA2 = PHI-180.
X2(1) = -LOC-LBP2*0.5*COSD(OMEGA2)
X2(2) = BEAM1*0.5-LBP2*0.5*SIND(OMEGA2)
W2 = 0.
V2(1) = VEL2*KTM*COSD(OMEGA2)
V2(2)=VEL2*KTM*SIND(OMEGA2)

END SUBROUTINE INI_COLLIS

A.3.5 collision.for

!!!** COLLISION.FOR  by Donghui Chen
!!!** VERSION: 2.1  DATE: December 10, 1999
!!!** REVISION NOTES:
!!!  v2.1
!!!   - Incorporate definite bow depth
!!!  v2.0
!!!   - Incorporate Double Hull Tankers
!!!   - MEMBRANE EFFECTS
!!!     - Consider friction
!!!     - Consider force to propagate yielding point
!!!  v1.2
!!!   - Adjusted in accordance with COLLISION_MOD.FOR and INITIATE.FOR
!!!  v1.1
!!!   - Separate out the initial part to INITIATE.FOR
!!!   - MEMBRANE EFFECTS
!!!     - Revised according to COLLISION_MOD.FOR
!!!  v1.0
!!!   - SIMULATION
!!!     - A new scheme for calculating accelerations is used in association
!!!       with the new method for calculating collision forces.
!!!     - Correct the formula of calculations of accelerations (Rev 03/04)
!!!     - Correct the formula of calculations of lost energy (Rev 03/04)
!!!** CONTENTS:
!!!  - SUBROUTINES
!!!    COLLISION - simulate the collision scenario of tankers
!!!
!!!** Subroutine COLLISION is to simulate the collision scenario of tankers
!!! Input : END_REL_TRANS - value of relevant transition to stop simulation
!!!           M - total number of scenarios (for testing output)
!!! Output: STAT - running status of the subroutine
!!!           0: successful; 1: LOC outside cargo tanks; 2: PEN >= BEAM;
!!!           3: PEN < 0
!!!           common blocks /STRUCK_COL/ and /STRIKE_COL/
!!!
!!! SUBROUTINE COLLISION(END_REL_TRANS, STAT, M)
!!!
!!! USE GENERAL
!!! USE SHIP_DIM
!!! USE SHIP_STR
!!! USE SHIP_COL
!!! USE COL_FORCE
!!! IMPLICIT NONE
!!!
!!! END_REL_TRANS - value of relevant transition to stop simulation
!!! STAT - running status
!!! INTEGER, OPTIONAL, INTENT (IN) :: M
!!! REAL, INTENT (IN) :: END_REL_TRANS
INTEGER, INTENT (OUT) :: STAT

INTEGER :: I, N=0

! STEP - the time step used in simulation
REAL :: STEP

! D_LOC - shift of impact point along the length during each time step
D_PEN - increment of penetration during each time step
DEFL - deflection of the shell/longitudinal bulkhead plate
REAL :: D_LOC, D_PEN
REAL, DIMENSION (2) :: DEFL

CP - current contact point
TRANS - transition of ships during each time step
REL-TRANS - relevant transition between two ships
ZETA - direction angle of relevant transition
ABS_REL_TRANS - absolute value of relevant transition
REAL, DIMENSION (2) :: CP, TRANS1, TRANS2, REL_TRANS
REAL :: ZETA, ABS_REL_TRANS

VAC - accelerations
VAC_MEM - accelerations caused by membrane effect
VAC_MIN - accelerations caused by Minorsky mechanism
WAC - angular speed acceleration
WAC_MEM - angular acceleration caused by membrane effect
WAC_MIN - angular acceleration caused by Minorsky mechanism
REAL, DIMENSION (2) :: VAC1, VAC2
REAL, DIMENSION (2) :: VAC1_MEM, VAC1_MIN, VAC2_MEM, VAC2_MIN
REAL :: WAC1, WAC2, WAC1_MEM, WAC1_MIN, WAC2_MEM, WAC2_MIN

F_MEM - reaction force caused by membrane effect
M_MIN - moment about origin of LOC-PEN system by membrane effect
F_MIN - reaction force caused by Minorsky mechanism
M_MIN - moment about origin of LOC-PEN system by Minorsky mechanism
REAL :: F_MEM, M_MEM, F_MIN, M_MIN, FML

ZETA1 - striking force angle in LOC-PEN system
ZETA2 - angle between the reaction force and striking ship principle axis
REAL :: ZETA1, ZETA2

ARM1 - length of the line from origin of LOC-PEN to X1
ARM2 - length of the line from origin of LOC-PEN to X1
ZETA1_LP - angle of the line from origin of LOC-PEN to X1 in LOC-PEN
ZETA2_LP - angle of the line from origin of LOC-PEN to X2 in LOC-PEN
XO_LP - coordinates of origin of LOC-PEN in global system
REAL :: ARM1, ARM2, ZETA1_LP, ZETA2_LP
REAL, DIMENSION (2) :: XO_LP

The following variables are for debugging purpose and output to damage.dat
ENERG - changes of kinetic energy
LOST_EN - total lost of kinetic energy
PHI - initial collision angle
LOC_INI - initial location of impact point
LOC_AFT - after end of damaged hull
LOC_FOR - fore end of damaged hull
PEN_MAX - maximum penetration
REAL :: ENERG, LOST_EN, PHI, LOC_INI, LOC_AFT, LOC_FOR, PEN_MAX

To initiate the variables
STAT=0
CALL INI_COLLIS
! for debugging
N=0
LOST_EN=0.
LOC_INI=LOC
PHI=OMEGA2+180.
LOC_AFT=LOC
LOC_FOR=LOC
PEN_MAX=0.
!
LBHD_CUR=1
DEFL=0.
REACHED=1
LBHD_RUP=0
LOC0(1)=LOC
LOC0(2)=0.
D_LOC0=0.
MEMB_EN0=0.
MINO_EN0=0.
DW0=0.
DF2=0.
OUTER=0.
INNER=0.
!
MV1=MASS_VIRT(A1, 90., DISP1)
MV2=MASS_VIRT(A2, 0., DISP2)
STEP=TIME_STEP(TDAB_THK, HEA, MV1(1), MV2(1))
ABS_REL_TRANS=V_ABS(V2-V1)*STEP
!
! virtual mass calculations for energy calculations of the first step
! for debug only (Rev 03/04)
MV1=MASS_VIRT(A1, OMEGA1, DISP1)
MV2=MASS_VIRT(A2, OMEGA2, DISP2)
!
OMEGAR(1)=OMEGA1-OMEGA2
BP1=PEN_BOW(LOC, PEN, OMEGAR(1), HEA, BEAM1)
XO_LP=</0., BEAM1*0.5/
ZETA1_LP=90.
ARM1=V_ABS(XO_LP)
ZETA2_LP=ATAN2D(X2(2)-XO_LP(2), X2(1)-XO_LP(1))
ARM2=V_ABS(X2-XO_LP)
!
!* Time domain simulation
DO WHILE (ABS_REL_TRANS>END_REL_TRANS)
N=N+1
!
!* To calculate the relevant transition between two ships at impact point
! current contact point
CP (1)=X2 (1)+LBP2*0.5*COSD(OMEGA2)
CP (2)=X2 (2)+LBP2*0.5*SIND(OMEGA2)
TRANS1 (1)=V1 (1)+(X1(2)-CP(2))*W1/RTD
TRANS1 (2)=V1 (2)-(X1(1)-CP(1))*W1/RTD
TRANS2 (1)=V2 (1)+(X2(2)-CP(2))*W2/RTD
TRANS2 (2)=V2 (2)-(X2(1)-CP(1))*W2/RTD
REL_TRANS=(TRANS2-TRANS1)*STEP
ABS_REL_TRANS=V_ABS(REL_TRANS)
IF (ABS_REL_TRANS==0.) THEN
EXIT
ELSE
ZETA=ATAN2D(REL_TRANS(2), REL_TRANS(1))
ENDIF
!
! To calculate the initial kinetic energy (for debug only) (Rev 03/04)
ENERG=.5*(MV1(1)*V1(1)**2+MV1(2)*V1(2)**2
& +MV2(1)*V2(1)**2+MV2(2)*V2(2)**2
& +2.*(MV1(3)*V1(1)*V1(2)+MV2(3)*V2(1)*V2(2))
& +(JV1*W1+JV2*W2)/(RTD*RTD))
!
! To calculate the shift of impact point and penetration during this step
D_PEN=ABS_REL_TRANS*SIND(OMEGA1-ZETA)
D_LOC=ABS_REL_TRANS*COSD(OMEGA1-ZETA)
IF (D_LOC0==0.) D_LOC0=D_LOC
!
FL_MAX=A1(1)*DISP1*D_LOC/(STEP*STEP)
!
! To determine the impact point and penetration after this step
LOC=LOC+D_LOC
PEN=PEN+D_PEN
!
IF (LBHD_CUR>LBHD_NUM.OR.PEN>=BEAM1) THEN
STAT=2
EXIT
ENDIF
IF (PEN<0.) THEN
STAT=3
EXIT
ENDIF
!
! To get the orientation of both ships
X1=X1+V1*STEP
X2=X2+V2*STEP
OMEGA1=OMEGA1+W1*STEP
OMEGA2=OMEGA2+W2*STEP
!
! To determine shape of the penetrated bow and damage extents
OMEGAR(2)=OMEGA1-OMEGA2
BP2=PEN_BOW(LOC, PEN, OMEGAR(2), HEA, BEAM1)
!
IF (PEN>PEN_MAX) PEN_MAX=PEN
IF (MIN(BP2(1,1), BP2(2,1), BP2(3,1))<LOC_FOR)
& LOC_FOR=MIN(BP2(1,1), BP2(2,1), BP2(3,1))
& IF (MAX(BP2(3,1), BP2(4,1), BP2(5,1))>LOC_AFT)
& LOC_AFT=MAX(BP2(3,1), BP2(4,1), BP2(5,1))
& IF (LOC_AFT>=TBHD_LOC(TBHD_NUM).OR.LOC_FOR<=TBHD_LOC(1)) THEN
LOC_AFT=TBHD_LOC(TBHD_NUM)
ELSE
LOC_FOR=TBHD_LOC(1)
ENDIF
STAT=1
EXIT
ENDIF
!
! virtual mass (Rev 03/04)
MV1=MASS_VIRT(A1, OMEGA1, DISP1)
MV2=MASS_VIRT(A2, OMEGA2, DISP2)
!
! To re-set the accelerations
VAC1_MEM=0.
VAC2_MEM=0.
WAC1_MEM=0.
WAC2_MEM=0.
VAC1_MIN=0.
VAC2_MIN=0.
WAC1_MIN=0.
WAC2_MIN=0.
VAC1=0.
WAC1=0.
VAC2=0.
WAC2=0.

!* To calculate the resistance force caused by Minorsky mechanism
CALL MINO_FORCE(D_LOC, D_PEN, F_MIN, ZETA1, M_MIN, M)
!
!* To calculate the acceleration of struck ship by Minorsky mechanism (Rev 03/04)
VAC1_MIN=ACC(F_MIN, ZETA1+OMEGA1-180., MV1)
WAC1_MIN=(-F_MIN*SIND(ZETA1-ZETA1_LP)*ARM1+M_MIN)/JV1
!
!* To calculate the acceleration of striking ship by Minorsky mechanism (Rev 03/04)
VAC2_MIN=ACC(F_MIN, ZETA1+OMEGA1, MV2)
WAC2_MIN=(F_MIN*SIND(ZETA2_LP-OMEGA1-ZETA1)*ARM2-M_MIN)/JV2
!
FML=F_MIN*COSD(ZETA1)
IF (FML*FL_MAX<0.) THEN
FL_MAX=FL_MAX-FML
ELSE
IF (ABS(FL_MAX)>ABS(FML)) THEN
FL_MAX=FL_MAX-FML
ELSE
FL_MAX=0.
END IF
ENDIF
!
!* To calculate the reaction forces of side and longitudinal bulkheads on both ships
!
!* To check whether reach the next longitudinal bulkhead
IF (PEN>BEAM1*0.5-LBHD_LOC(LBHD_CUR+REACHED).AND. &
LBHD_RUP(LBHD_CUR+REACHED)<2) THEN
REACHED=REACHED+1
LOC0(REACHED)=LOC
ENDIF
!
!* To calculate the deflection at this step
DO I=1, REACHED
DEFL(I)=PEN-BEAM1*0.5+LBHD_LOC(LBHD_CUR+I-1)
IF (DEFL(I)<0.) THEN
REACHED=I-1
EXIT
ENDIF
END DO
!
!* To calculate the force of membrane and web frames
CALL MEMB_FORCE(LOC, D_LOC, D_PEN, DEFL, F_MEM, ZETA1, M_MEM, &
M)
!
!* To calculate the acceleration of struck ship (Rev 03/04)
!* Note: the membrance force here is to include friction force (Rev. 09/15)
VAC1_MEM=ACC(F_MEM, ZETA1+OMEGA1-180., MV1)
WAC1_MEM=(-F_MEM*SIND(ZETA1-ZETA1_LP)*ARM1+M_MEM)/JV1
!
!* To calculate the acceleration of striking ship (Rev 03/04)
VAC2_MEM=ACC(F_MEM, ZETA1+OMEGA1, MV2)
WAC2_MEM=(F_MEM*SIND(ZETA2_LP-OMEGA1-ZETA1)*ARM2-M_MEM)/JV2
!
!* To get the accelerations for this time step
VAC1=VAC1_MEM+VAC1_MIN
VAC2=VAC2_MEM+VAC2_MIN
WAC1=(WAC1_MEM+WAC1_MIN)*RTD
WAC2=(WAC2_MEM+WAC2_MIN)*RTD

!* To calculate the new velocity
V1=V1+VAC1*STEP
V2=V2+VAC2*STEP
W1=W1+WAC1*STEP
W2=W2+WAC2*STEP

XO_LP(1)=X1(1)+ARM1*COSD(OMEGA1+ZETA1_LP)
XO_LP(2)=X1(2)+ARM1*SIND(OMEGA1+ZETA1_LP)
ZETA2_LP=ATAN2D(X2(2)-XO_LP(2), X2(1)-XO_LP(1))-OMEGA1
ARM2=V_ABS(X2-XO_LP)
OMEGAR(1)=OMEGAR(2)
BP1=BP2

!* To calculate the changes of kinetic energy (for debug only) (Rev 03/04)
ENERG=ENERG-.5*(MV1(1)*V1(1)**2+MV1(2)*V1(2)**2
&+MV2(1)*V2(1)**2+MV2(2)*V2(2)**2
&+2.*(MV1(3)*V1(1)*V1(2)+MV2(3)*V2(1)*V2(2))
&+(JV1*W1+JV2*W2)/(RTD*RTD))
LOST_EN=LOST_EN+ENERG

!* test print
IF (PRESENT(M)) THEN
  IF (M<3) THEN
    WRITE (23,'(I4, 2F20.1, /)') N, ENERG, LOST_EN
    WRITE (22,'(/, I4, 6(4X, F10.5))') N, PEN, D_PEN, LOC,
      & D_LOC, ABS_REL_TRANS, ZETA
    WRITE (22,'(4X, 6(4X, F10.5))') V1, W1, V2, W2
    WRITE (22,'(4X, 6(4X, F10.5))') X1, OMEGA1, X2, OMEGA2
  END IF
END IF
END IF

IF (F_MIN<1.E-2) EXIT
IF (ENERG<0.) EXIT

END DO

WRITE (21,'(1X,4(4X,F8.3))') LOC_INI, PHI, PEN_MAX,
& LOC_AFT-LOC_FOR
WRITE (24,'(1X,4(4X,F8.3))') LOC_INI, PHI, OUTER, INNER

END SUBROUTINE COLLISION